Elektronický manuál pro program SCILAB

Martina Vaculíková

Bakalářská práce
2006

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2005/2006

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Martina VACULÍKOVÁ**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Elektronický manuál pro program SCILAB**

Zásady pro vypracování:

1. Proveďte literární rešerši na zadané téma.
2. V teoretické části práce popište základy ovládání programového prostředí SCILAB, zadávání příkazů a možnosti programu v oblasti práce s maticemi a oblasti vizualizace výsledků.
3. Vytvořte elektronický manuál k programovému prostředí SCILAB ve formě HTML stránek. Manuál bude sloužit pro výukové účely a bude obsahovat popis ovládání programu, základy práce s programem, rozdělení příkazů do jednotlivých skupin.
4. U jednotlivých vybraných příkazů z daných skupin uveďte jejich podrobný popis a jejich použití demonstrujte na vhodně zvolených příkladech.

Rozsah práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

INRIA. Scilab : A Free Scientific Software Package [online]. c1989-2006 , last updated: October 04 2005 [cit. 2006-01-12]. Dostupný z WWW: <http://www.scilab.org/>.
SCILAB GROUP. Introduction to Scilab : User's Guide. [online]. c1989-2006 [cit. 2006-01-12]. Dostupný z WWW: <http://www.scilab.org/doc/intro/intro.pdf>.
Scilab Group. Scilab demonstration pages [online]. c1989-2006 , last updated: December 20 2005 [cit. 2006-01-12]. Dostupný z WWW: <http://www.scilab.org/doc/demos_html/index.html>.
Scilab Documentation (4.0-RC1 version) [online]. c1989-2006 [cit. 2006-01-12]. Dostupný z WWW: <http://www.scilab.org/product/man-eng/index.html>.

Vedoucí bakalářské práce: **Ing. Petr Navrátil**
Ústav řízení procesů

Datum zadání bakalářské práce: **14. února 2006**

Termín odevzdání bakalářské práce: **16. června 2006**

Ve Zlíně dne 14. února 2006

prof. Ing. Vladimír Vašek, CSc.
pověřený děkan

L.S.

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Cílem bakalářské práce je vypracování elektronického manuálu pro program Scilab v prostředí HTML stránek. Scilab je vědecký matematický program pro numerické výpočty, který obsahuje množství toolboxů a umožňuje pracovat s vyššími programovacími jazyky. Je to zdarma dostupný program, který díky své vysoké kvalitě dokáže konkurovat i drahým programům. V manuálu je podrobně popsána práce v uživatelském prostředí. Použití funkcí je ukázáno na příkladech a grafech.

Klíčová slova: Scilab, manuál, numerické výpočty, vizualizace, programování

ABSTRACT

The aim of this final work is create an electronic manual for Scilab program in the environment of HTML pages. Scilab is scientific software package for numerical computations include number of toolboxes and provide work with high-level programming languages. It’s freely distributed program, which can compete to other expensive programs. Work in the environment is fully described in the manual. Application of functions is demonstrated on many examples and figures.

Keywords: Scilab, user's guide, numerical computations, visualization, programming

Ráda bych poděkovala vedoucímu mé bakalářské práce Ing. Petru Navrátilovi za odborné vedení, za připomínky a čas věnovaný mé práci.

Ve Zlíně, 23. 05. 2006 .....

Martina VACULÍKOVÁ

OBSAH

ÚVOD

Na trhu software existuje mnoho programů pro numerické výpočty. Většina z nich jsou velmi kvalitní balíky, obsahující množství přídavných modulů, tzv. toolboxů.

Matematické programy jako Matlab, Maple nebo Mathematica poskytují široké spektrum využití pro symbolické i numerické výpočty, 2D a 3D grafiku či simulaci. Tyto programy ovšem nejsou zdarma dostupné, proto mnoho uživatelů hledá jinou alternativu mezi volně dostupnými programy, které by svou kvalitou dokázaly konkurovat i drahým programům.

Takovým programem je i Scilab od francouzských firem INRIA (The French National Institute for Research in Computer Science and Control) a ENPC (École Nationale des Ponts et Chaussées).

Scilab je volně šiřitelný matematický program pro numerické výpočty, který obsahuje množství toolboxů a umožňuje pracovat s vyššími programovacími jazyky.

Velkou výhodnou tohoto programu je to, že je zdarma, proto si ho může stáhnout téměř každý. I když byl Scilab původně vyvíjen pro operační systém Linux, dnes je dostupný i pro další operační systémy (Windows, Solaris).

Cílem této bakalářské práce je vytvoření elektronického manuálu, kde je podrobně popsána práce v uživatelském prostředí, základní operace s vektory, maticemi, polynomy a elementárními matematickými funkcemi. Nejvíce prostoru je věnováno grafickému prostředí a vizualizaci. Nakonec jsou vysvětleny základy programování, práce se soubory a funkce jazyka C, dále vytváření vlastních funkcí a skriptů.

Manuál je tematicky rozdělen do sedmi kapitol. V každé kapitole je stručný úvod do dané problematiky, dále popis funkcí rozčleněný do podkapitol, a na konci kapitol je vždy seznam funkcí a seznam příkladů. U každé funkce je podrobně vysvětlena její syntaxe a použití je demonstrováno na příkladech a grafech.

I. TEORETICKÁ ČÁST

1 CO JE SCILAB

Scilab je vědecký softwarový balík pro numerické výpočty. Poskytuje otevřené programovací prostředí pro inženýrské a vědecké aplikace.

Scilab byl vyvíjen od roku 1990 firmami **INRIA** (The French National Institute for Research in Computer Science and Control) a **ENPC** (École Nationale des Ponts et Chaussées). Od roku 1994 byl distribuován jako freeware i se zdrojovým kódem. V současné době je používán pro výukové a průmyslové prostředí po celém světě. Scilab nyní spravuje Scilab Consorcium, založené v květnu 2003.

Scilab obsahuje stovky matematických funkcí s možností přidat další programy z různých programovacích jazyků (FORTRAN, C, C++, JAVA...). Má propracovanou strukturu dat, překladač, a dovoluje používat vyšší programovací jazyky. Scilab byl vyvinut s myšlenkou, že bude volně šiřitelný.

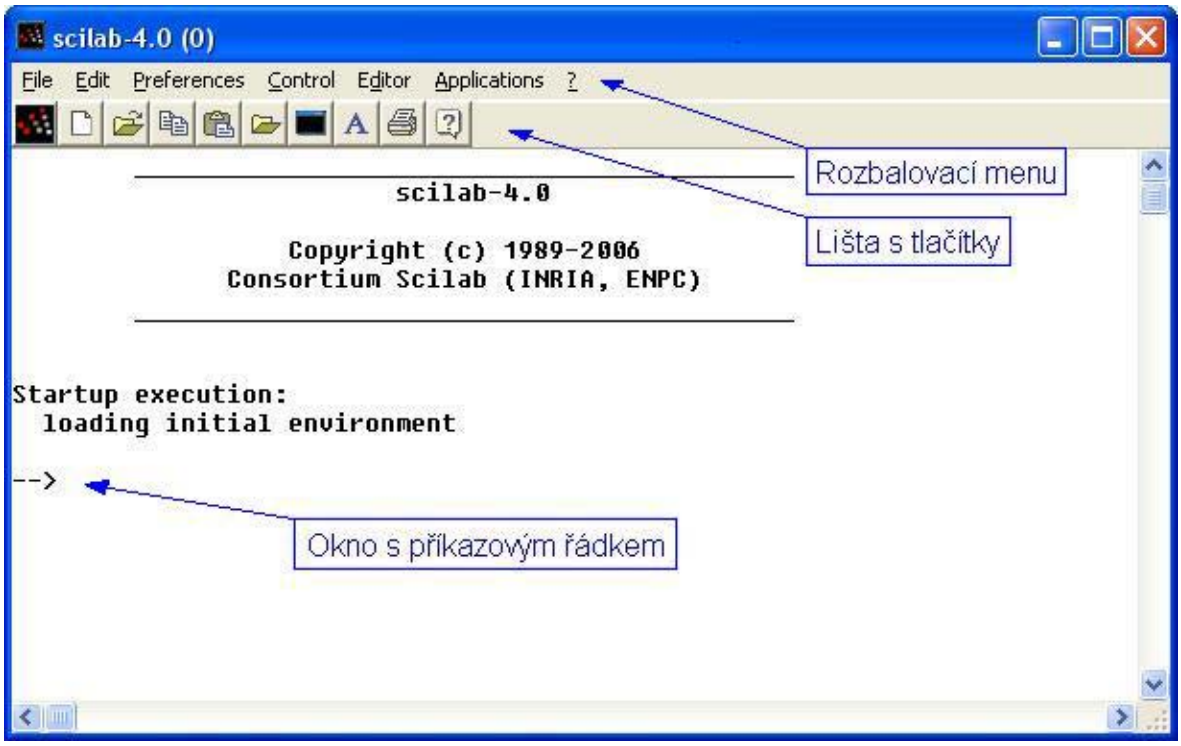
Scilab zahrnuje množství toolboxů:

- 2D a 3D grafika, animace, grafy
- lineární algebra, matice
- polynomiální a logické funkce
- simulace: řešitel diferenciálních rovnic
- Scicos: modelář a simulátor hybridních dynamických systémů
- klasické a robustní řízení, LMI optimalizace
- diferencovatelná a nediferencovatelná optimalizace
- řízení signálů
- paralelní Scilab využívající PVM
- statistika
- prostředí počítačové algebry (Maple, MuPAD)
- prostředí s TCL/TK

Scilab dokáže pracovat pod operačními systémy Windows 9X/2000/XP, GNU/Linux a UNIX. Zdrojové kódy jsou volně ke stažení na domovské stránce www.scilab.org. [1]

2 PROSTŘEDÍ SCILABU

Po spuštění programu se objeví následující okno:



Obrázek 1: Okno Scilabu

Hlavní část okna je tvořena příkazovým řádkem, kam se píší všechny příkazy. V horní části okna je rozbalovací menu s mnoha funkcemi a nástrojová lišta pro rychlé spouštění nebo editaci.

Každý příkaz se po stisku klávesy Enter ihned provede. Očekává-li Scilab příkaz, pak jsou na začátku řádku úvodní znaky -->. Jestliže v příkazu neuvedeme název proměnné, do které se má výsledek uložit, uloží se do proměnné s názvem ans (answer). Pokud příkaz ukončíme středníkem (;), příkaz se provede, ale výsledek se nevypíše. Pokud příkaz neukončíme středníkem, výsledek se i vypíše.

Scilab také rozlišuje malá a velká písmena (case-sensitiv), takže Ahoj, ahoj a AHOJ je interpretováno jako tři různé objekty.

Komentáře se píší za dvojitým lomítkem // a končí na konci řádku. Cokoli napsaného za // je bráno jako komentář. Komentář nesmí začínat sekvencí //end. Při běhu programu jsou komentáře ignorovány.

2.1 Příkazový řádek

Po spuštění programu Scilab se v hlavním okně zobrazí banner s informacemi o programu a pod ním příkazový řádek (prompt) uvozený značkami -->. Příkazy jsou zadávány přímo z klávesnice. Po příkazovém řádku se lze pohybovat kurzorovými šipkami doprava nebo doleva, šipkami nahoru a dolů se vyvolají poslední zadané příkazy.

Dalšími příkazy pro editaci řádku jsou:

Tabulka I: Seznam klávesových zkratek pro editaci řádku

Zkratka	Funkce
ctrl+P	předchozí příkaz (previous command)
ctrl+N	další příkaz (next command)
ctrl+B	zpět o jeden znak(back one character)
ctrl+F	vpřed o jeden znak(forward one character)
ctrl+A	na začátek řádku(begging of the line)
ctrl+E	na konec řádku(end of the line)
ctrl+H	smazat předchozí znak(delete previous character)
ctrl+D	smazat aktuální znak(delete current character)
ctrl+W	smazat minulé slovo(delete last word)
ctrl+K	smazat do konce řádku(delete to end of line)
ctrl+U	smazat celý řádek(delete entire line)

2.2 Nápověda

Nápovědu lze vyvolat příkazem **help** v příkazovém řádku, klávesou F1 nebo v okně rozbalovacího menu v nabídce ? odkaz Scilab help. V levém sloupci nápovědy je seznam všech témat. Po kliknutí na libovolné téma se rozbalí seznam příkazů k vybranému tématu.

V pravém sloupci se zobrazí podrobné informace k vybrané položce: jednoduchá definice příkazu, vyvolávací sekvence, syntaxe příkazu, parametry, podrobnější popis příkazu, příklady a odkazy na podobné příkazy.

Nápověda je rozčleněna do tématických skupin. Příkazy nejsou ve skupinách rozčleněny systematicky, ale podle abecedy.

2.3 Proměnné a funkce

Proměnné mohou být až 24 znaků dlouhé (další znaky program ignoruje). Název proměnné začíná písmenem nebo znaky %, _, #, !, $, ?. Scilab rozlišuje malá a velká písmena (*case-sensitiv*). Pro vypsání všech aktuálně používaných proměnných slouží příkaz **who**. Pokud si nadefinujeme vlastní proměnné, program nám je také vypíše. Příkazem **whos** dostaneme podrobný seznam všech proměnných, který obsahuje název proměnné, typ proměnné, rozměr a velikost v bytech. Většina z těchto proměnných jsou systémové proměnné nebo speciální konstanty, jejichž obsah nemůže uživatel změnit. Seznam uživatelem definovaných proměnných lze také získat příkazem **browsevar** nebo spuštěním z nástrojové lišty - Applications -> Browser Variables. Výpis proměnných se otevře v novém okně *ScilabBrowseVar*. Tento prohlížeč umožňuje pracovat s uživatelem definovanými proměnnými v prostředí *TCL/TK*. Po otevření okna dostaneme podrobný seznam proměnných.

Funkce je element jazyka, který po zavolání vrací hodnotu nebo daný typ a také může vyvolat nějakou akci. Uživatel si může definovat vlastní funkce a doplnit Scilab svou vlastní knihovnou. Funkce se liší jménem, ale také počtem a typem argumentů.

2.4 Speciální konstanty

Tyto speciální konstanty jsou předdefinované, jsou chráněné a nelze je smazat.

Tabulka II: Speciální konstanty

Název	Hodnota
%i	imaginární jednotka (i^2=-1)
%pi	p = 3.1415927
%e	základ přirozených logaritmů e = 2.7182818
%eps	maximální hodnota, pro kterou platí 1 + %eps = 1 (eps = 2.22 . 10^{-16})
%inf	infinity (nekonečno)
%nan	not-a-number
%t	true (pravda)
%f	false (nepravda)

3 ZÁKLADNÍ PRÁCE S MATICEMI

Teorie matic a determinantů představuje základ lineární algebry. Nejrozsáhlejší využití mají matice a determinanty při řešení systémů lineárních rovnic. Zakladatelem teorie matic je anglický matematik A. Cayley (1821-1895). Pojem determinantu zavedl v roce 1693 německý matematik W. G. Leibniz (1646-1716). Jeho objev byl ale zapomenut. O znovuobjevení pojmu determinant se později zasloužil švýcarský matematik G. Cramer (1704-1752). Na dalším rozvoji teorie matic se podíleli zejména G. Frobenius (1849-1917), J. J. Sylvester (1814-1897) a K. Weierstrass (1815-1897). [11]

Ve Scilabu je matice definována jako obdélníkové pole typu (*m,n*) obsahující data stejného typu (boolean, float, integer, string, polynom,...) uspořádaných do *m* řádků a *n* sloupců. Matice *A* typu (*m,n*) má potom obecný tvar

$$M = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \qquad (1)$$

Jednořádkovou maticí, tedy maticí typu (*m,1*), je řádkový vektor. Jednosloupcovou maticí, tedy maticí typu (*1,n*), je sloupcový vektor. Matice typu (*1,1*) je skalár.

3.1 Vytváření vektorů a matic

Vektor můžeme definovat jako jednorozměrnou matici. Existují dva typy vektorů - řádkové a sloupcové vektory.

Vektory a matice se vkládají mezi hranaté závorky [ ]. Prvky v řádku matice se oddělují čárkou nebo mezerou, prvky ve sloupci se oddělují středníkem.

Matici $M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ vytvoříme ve Scilabu příkazem

```
--> M = [1 2 3; 4 5 6; 7 8 9]
```

Vektor tvořený aritmetickou řadou (například od 0 do 10 s krokem 0.5) si můžeme vygenerovat automaticky. Ten má tvar:

vektor = počáteční hodnota : krok : koncová hodnota.

Pokud není zadána hodnota kroku, výchozí hodnota je 1. Jestliže chceme sestupnou řadu čísel, jako krok musíme zadat zápornou hodnotu. Například pro vygenerování řady čísel od 1 do 10 s krokem 0.5 se zadá příkaz

```
--> x = 1 : 0.5 : 10
```

Vektory vytvořené tímto způsobem jsou řádkové vektory.

Pro vytvoření matice náhodných čísel můžeme použít funkci **rand,** která generuje náhodná čísla z intervalu <0;1>. Funkce **rand(m, n)** má parametry *m,n,* kde *m* je počet řádků a *n* počet sloupců matice.

Například vytvoříme matici náhodných čísel z intervalu <0;100>:

```
--> M = rand (2,3)*100
```

$$M = \begin{bmatrix} 36.16361 & 56.642488 & 33.217189 \\ 29.22266 & 0.0221135 & 93.296162 \end{bmatrix}$$

3.2 Základní operace s maticemi a vektory

Základními maticovými operátory jsou:

Tabulka III: Maticové operátory

Operátor	Význam
[ ]	definice matice, zřetězení
;	oddělovač řádků
( )	pro práci s prvky matice
’	transpozice
+	součet
-	rozdíl
*	násobení
\	levé dělení
/	pravé dělení
^	exponent
.*	násobení jednotlivých prvků
.\	levé dělení jednotlivých prvků
./	pravé dělení jednotlivých prvků

.^	exponent jednotlivých prvků
.*.	násobení každého prvku s každým

3.2.1 Transpozice

Transponovaný vektor je vektor, který doplníme na čtvercovou matici, ta se otočí kolem hlavní diagonály a odebere se přidaná část. To znamená, že z řádkového vektoru se stane sloupcový a naopak. Transponovaná matice je matice, kterou doplníme na čtvercovou matici, ta se otočí kolem hlavní diagonály a odebere se přidaná část. To znamená, že z matice 3x2 se stane matice 2x3. Pro transpozici se používá znak '. [12]

```
--> v = [1 2 3]; v'
```

$$ans = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

```
--> A = [1 2 3; 4 5 6; 1 -1 0]; A'
```

$$ans = \begin{bmatrix} 1 & 4 & 1 \\ 2 & 5 & -1 \\ 3 & 6 & 0 \end{bmatrix}$$

3.2.2 Aritmetické operace

Sčítat (a odčítat) můžeme jen matice nebo vektory stejného rozměru. Při sčítání (odčítání) se vždy sečtou (odečtou) prvky matice nebo vektoru se stejným indexem.

Nadefinujeme dvě matice *A* a *B*, které sečteme a odečteme:

```
--> A = [1 2 3; 4 5 6; 1 -1 0]
```

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 1 & -1 & 0 \end{bmatrix}$$

```
--> B = [1 1 1; 2 3 -1; 5 7 -2]
```

$$B = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 3 & -1 \\ 5 & 7 & -2 \end{bmatrix}$$

```
--> A + B
```

$$ans = \begin{bmatrix} 2 & 3 & 4 \\ 6 & 8 & 5 \\ 6 & 6 & -2 \end{bmatrix}$$

```
--> A - B
```

$$ans = \begin{bmatrix} 0 & 1 & 2 \\ 2 & 2 & 7 \\ -4 & -8 & 2 \end{bmatrix}$$

Matici rozměrů $M \times N$ můžeme vynásobit libovolnou maticí rozměrů $N \times P$, přičemž výsledkem je matice o rozměrech $M \times P$. Násobení probíhá tak, že každé číslo výsledné matice získáme jako skalární součin příslušného řádku první matice s příslušným sloupcem druhé matice.

```
--> A * B
```

$$ans = \begin{bmatrix} 20 & 28 & -7 \\ 44 & 61 & -13 \\ -1 & -2 & 2 \end{bmatrix}$$

Při násobení vektorů můžeme násobit řádkový a transponovaný vektor – výsledkem je skalár. Při násobení transponovaného a řádkového vektoru je výsledkem matice.

```
--> u = [-1 2 3]; v = [6 -10 0];
--> u * v'
```

ans = -26

```
--> u' * v
```

$$ans = \begin{bmatrix} -6 & 10 & 0 \\ 12 & -20 & 0 \\ 18 & -30 & 0 \end{bmatrix}$$

Pokud chceme násobit jednotlivé prvky vektoru, musíme použít operátor "." a příslušnou funkci :

```
--> u. * v
```

$$ans = \begin{bmatrix} -6 & -20 & 0 \end{bmatrix}$$

3.2.3 Funkce sum

Funkce **sum(x)** udělá součet všech prvků vektoru nebo matice. Funkce *y = sum(x,'r')* (nebo analogicky *y = sum(x,1)*) je součet prvků ve sloupci *y(j) = sum(x (:, j))*. Výsledkem je

řádkový vektor. Funkce *y = sum(x,'c')* (nebo analogicky *y = sum(x,2))* je součet prvků v řádku *y(i) = sum(x(i,:))* . Výsledkem je sloupcový vektor.

Příklad: součet čísel od 1 do 100. Nejprve si vytvoříme vektor s čísly od 1 do 100, a pak funkcí sum sečteme všechny čísla.

```
--> v = 1:100; sum (v)
```

ans = 5050

3.2.4 Funkce diag

Pro matici *m* funkce **diag(m)** vypíše prvky ležící na hlavní diagonále. Pro definovaný vektor pak funkcí **diag(v)** vytvoříme z řádkového nebo sloupcového vektoru matici, na jejíž hlavní diagonále budou ležet prvky vektoru.

```
--> v = [1 2 3 4];  diag(v)
```

$$ans = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix}$$

3.2.5 Funkce size

Funkce **size** určí velikost (rozměr) matice. Je definována jako **rozmer = size (matice)**, přičemž výsledkem je vektor o rozměru 1x2 (počet řádků, počet sloupců).

Výstup si můžeme definovat také pomocí proměnných. Syntaxe potom vypadá takto :

[pocet_radku,pocet_sloupcu] = size(vektor).

Příklad : nadefinujme matici *M* o rozměru 3x2 a pomocí funkce *size* ověříme její rozměr

```
--> M = [1 2; 3 4; 5 6]
--> [radku, sloupcu] = size(M)
```

sloupcu = 2

radku = 3

3.2.6 Funkce linspace a logspace

Funkce **linspace** slouží pro vytváření vektoru, ve kterém jsou jeho prvky lineárně rozmístěné. Jinak řečeno, funkce *linspace* (x_1, x_2, *n*) vytvoří řádkový vektor o *n* prvcích

(přednastavená hodnota je 100), jehož body jsou rovnoměrně lineárně rozprostřené mezi x_1 a x_2.

Příklad: chceme vektor s čísly od 0 do 100 s krokem 10. Na výběr máme dvě možnosti řešení: vektor můžeme automaticky vygenerovat příkazem *x = 0:10:100*, nebo použijeme funkci *linspace*.

```
--> v = (0, 100, 11)
```

v = 0 10 20 30 40 50 60 70 80 90 100

Velmi podobná funkci **linspace** je funkce **logspace**. Ta vygeneruje prvky vektoru logaritmicky rozmístěné mezi body 10^{d1} a 10^{d2}.

Syntaxe této funkce je **logspace(d1,d2, n),** kde *d1* a *d2* jsou počáteční a koncová hodnota, *n* je počet hodnot (přednastavená hodnota 50).

Příklad: vygenerujeme logaritmickou řadu od 10^1 do 10^5

```
--> v = logspace(1,5,5)
```

v = 10 100 1000 10000 100000

3.2.7 Další funkce pro práci s maticemi

Funkce det

Funkce **det(x)** vypočítá determinant matice. Determinant je číslo přiřazené čtvercové matici (pro jiné matice není determinant definován), které tuto matici nějakým způsobem charakterizuje. Vytvoříme si například matici *D* a určíme její determinant :

```
--> D = [2 -3 8; 4 6 -7; -5 4 -9];
--> det(D)
```

ans = 103

Funkce inv

Funkce **inv(x)** vypočítá inverzní matici. Inverze matice je jakousi obdobou dělení. Inverzní matice je definována pouze pro čtvercové matice, pro jiné nemá smysl. Pokud matici vynásobíme její inverzní maticí, získáme jednotkovou matici.

Matematická definice: $B=A^{-1}$, $A \times B=B \times A=E$, kde A,B jsou čtvercové matice, E je jednotková matice.

Funkce rank

Funkce **rank** určí hodnost matice. Hodnost matice je rovna maximálnímu počtu jejích lineárně nezávislých řádků.

Funkce trace

Funkce **trace** provede součet čísel ležících na hlavní diagonále. Toto číslo se nazývá stopa matice. Příkaz *trace(A)* je to samé jako *sum (diag(A))*.

Funkce eye, ones, zeros

Funkce **eye** vytvoří jednotkovou matici, tj. matici s jedničkami na hlavní diagonále. Funkce **ones** naplní matici jedničkami. Funkce **zeros** naplní matici nulami.

3.2.8 Indexování matic

Mějme matici *M*. Potom prvek v řádku *i* a sloupci *j* je vyjádřen jako **M (i,j).** Například *M(3,2)* je prvek ležící ve třetím řádku a druhém sloupci. K jednotlivým prvkům matice také můžeme přistupovat přes jediný index **M(k).** V případě M(3,2) = M(7), počítáno po sloupcích. Přístup k více prvkům umožňuje operátor dvojtečka **:**. Zápis **M(1:m,n)** vybere prvky v řádku *1* až *m* ve sloupci *n* v matici *M*. Mazání řádků a sloupců se provádí použitím dvojice prázdných hranatých závorek. Jen jeden prvek nelze smazat, protože výsledkem této operace by nebyla matice a program by vypsal chybové hlášení.

Jako příklad vytvoříme matici *M* o velikosti 4×4:

```
--> M = [1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16];
```

$$ans = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

Přístup k prvku *(3,2)* realizujeme zápisem *M(3,2)* nebo ekvivalentně *M(7):*

```
--> M(3,2)
```

$ans = 10$

Výpis druhého, třetího a čtvrtého prvku ve třetím sloupci provedeme zápisem

```
--> M(2:4,3)
```

$$ans = \begin{bmatrix} 7 \\ 11 \\ 15 \end{bmatrix}$$

Pro smazání druhého sloupce matice použijeme příkaz

```
--> M(:,2) = []
```

$$ans = \begin{bmatrix} 1 & 3 & 4 \\ 5 & 7 & 8 \\ 9 & 11 & 12 \\ 13 & 15 & 16 \end{bmatrix}$$

3.3 Polynomy

Scilab poskytuje jednoduché vytváření a práci s polynomy (mnohočleny). Polynomy jsou prezentovány jako vektory. Můžeme vytvářet polynomiální matice, práce s nimi je potom velmi podobná práci s číselnými maticemi.

3.3.1 Definice polynomu

Pro definování polynomu existuje příkaz **poly**. Máme dvě možnosti, jak polynom vytvořit : specifikovat koeficienty polynomu nebo kořeny polynomu.

Základní syntaxe: **[p]=poly(a, "x", ["flag"])**, kde *a* je matice, vektor nebo číslo, *x* je proměnná a *flag* je řetězec, který má hodnotu *"roots"* nebo *"coeff"* podle toho, jak chceme polynom definovat, přednastavená hodnota je *"roots"*. Koeficient u nejnižší mocniny je uložen v prvním prvku vektoru a další koeficienty postupně patří k vyšším mocninám.

Nejprve tedy vytvoříme polynom *p* zadáním koeficientů :

```
--> p = poly([1 2 3],"x","coeff")
```

$p = 1 + 2x^2 + 3x$

Koeficienty polynomu můžeme také nadefinovat předem a uložit do vektoru

```
--> v = [5, 4, 3, 2, 1];
--> w = poly (v, "y","coeff")
```

$$w = 5 + 4y^2 + 3y^3 + 2y^4 + y$$

Zadáním kořenů polynomu vytvoříme polynom q :

```
--> q = poly([1 2],"x")
```

$$q = 2 - 3x^2 + x$$

Další možností vytváření polynomů je první definovat samotnou proměnnou a potom polynom zapsat v algebraické formě.

```
--> x = poly (0, "x");
--> p=3*x^2+4*x+1
```

$$p = 1 + 4x + 3x^2$$

3.3.2 Základní operace s polynomy

Mezi základní matematické operace s polynomy patří sčítání, odčítání, násobení a dělení. Při počítání si musíme dát pozor na to, aby polynomy měly shodnou proměnnou, jinak by program ohlásil chybu. Při dělení polynomu polynomem vznikne lomený výraz.

Příklad: máme polynomy *p* a *q*, tyto polynomy sečteme, vynásobíme a nakonec vydělíme:

```
--> x = poly (0, "x");
--> p=3*x^2+4*x+1;
--> q=x^2+5*x-6;
```

Součet polynomů:

```
--> p+q
```

$$ans = -5 + 9x + 4x^2$$

Násobení polynomů:

```
--> p*q
```

$$ans = -6 - 19x + 3x^2 + 19x^3 + 3x^4$$

Dělení polynomů - vznikne lomený výraz:

```
--> p/q
```

$$ans = \frac{1 + 4x + 3x^2}{-6 + 5x + x^2}$$

3.3.3 Kořeny polynomu

Máme nějaký polynom *p* a chceme znát jeho kořenové činitele. Ve Scilabu existuje funkce **roots**, která ze zadaného polynomu určí jeho kořeny.

Syntaxe této funkce je **y = roots(p),** kde *p* je polynom a *y* je vektor hledaných kořenů.
Příklad: definujeme polynom *p* a určíme jeho kořeny:

```
--> x = poly (0, "x");
--> p = x^4 - 15*x^2 + 10*x + 24
```

$$p = 24 + 10x^2 - 15x + x^4$$

--> y=roots(p)

$$y = \begin{matrix} -1 \\ 2 \\ 3 \\ -4 \end{matrix}$$

3.3.4 Charakteristický polynom matice

Charakteristický polynom je definován jako *determinant(x*I - M),* kde *M* je matice, x je proměnná a *I* je jednotková matice (*eye*). V tomto případě musí být první argument funkce *poly* čtvercová matice.

Výpočet charakteristického polynomu:

```
--> poly([1 2;3 4],"y","roots")
```

$$ans = -2 - 5y + y^2$$

3.3.5 Polynomiální matice

Polynomy můžeme také použít jako prvky matice, pak vytvoříme polynomiální matici.

```
--> P = [1/s  1/(1+s); 1/(1+s)  1/s^2]
```

$$P = \begin{matrix} \frac{1}{s} & \frac{1}{1+s} \\ \frac{1}{1+s} & \frac{1}{s^2} \end{matrix}$$

4 ELEMENTÁRNÍ FUNKCE

V této kapitole se budeme věnovat základnímu popisu elementárních matematických funkcí a jejich použití ve Scilabu. Mezi tyto funkce zařadíme jednoduché matematické funkce pro práci s odmocninami, absolutní hodnotou nebo zaokrouhlování, exponenciální a logaritmické funkce a nakonec goniometrické funkce.

Funkci ve tvaru *y = f(x)* graficky znázorníme příkazem *plot* nebo *plot2d*. Tyto funkce můžeme použít pro vykreslení grafů matematických funkcí. Obě funkce jsou podrobně popsány v další kapitole.

4.1 Funkce pro práci s čísly

4.1.1 Zaokrouhlování

Pro zaokrouhlování existuje ve Scilabu několik funkcí. Funkce **ceil** zaokrouhluje nahoru k nejbližšímu celému číslu, funkce **floor** dolů k celému číslu. Funkce **round** zaokrouhluje podle matematických pravidel. Funkce **fix** číslo za desetinnou čárkou zaokrouhlí k nule. Funkce **int** vrací celou část čísla.(*integer*). Funkce **floor** a **fix** zaokrouhlují dolů k celému číslu, jejich výsledek tedy bude stejný s funkcí **int**.

Všechny tyto funkce pro zaokrouhlování můžeme použít i pro **komplexní čísla**, přičemž funkce jsou aplikovány jednotlivě na reálnou a imaginární část.

4.1.2 Funkce clean

Funkce **clean (A, [epsa])** zaokrouhlí k nule velmi malá čísla pro absolutní hodnoty menší než parametr *epsa*. Tato funkce může být užitečná pro „vyčištění“ matic. Při násobení matice její inverzní maticí vznikne jednotková matice. Ve Scilabu díky drobným numerickým chybám při výpočtu inverzní matice vznikne matice s velmi malými čísly. Funkce *clean* tato velmi malá čísla „vyčistí“ (zaokrouhlí na nulu) a tak dostaneme požadovanou jednotkovou matici.

Pro ukázku vytvoříme matici *A* o velikosti 3×3, vynásobíme ji její inverzní maticí a vzniklou matici *B* vyčistíme příkazem *clean*.

```
--> A = [1 3 2;-1 2 1;4 2 1]
--> B = A*inv(A)
```

$$ans = \begin{bmatrix} 1 & -3.053D-16 & -8.327D-17 \\ -2.776D-17 & 1 & -2.776D-17 \\ 1.110D-16 & -3.331D-16 & 1 \end{bmatrix}$$

Tato velmi malá čísla zaokrouhlíme k nule pomocí funkce *clean*:

```
--> B = clean(B)
```

$$ans = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4.1.3 Funkce abs

Funkce **abs** vrací absolutní hodnotu čísla. U komplexních čísel je absolutní hodnota definována jako $|z| = sqrt\ (a^2 + b^2)$.

4.1.4 Funkce sign

Funkce **sign(A)** vrací hodnotu 1, -1 nebo 0 pro reálná čísla podle toho jestli je číslo kladné, záporné nebo nula. Nazývá se také znaménková funkce. Pro komplexní čísla vrací funkce *sign* hodnotu $\frac{z}{|z|}$.

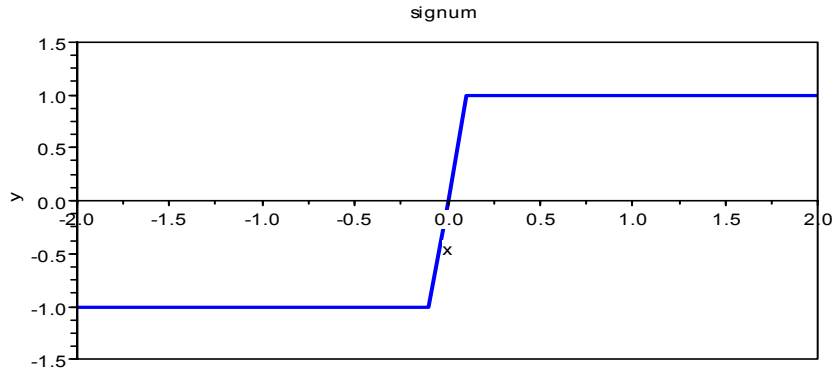
Příklad funkce *sign* pro reálná i komplexní čísla:

```
--> B = sign([-2.4 0.0 5.8])
```

$$ans = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

```
--> B = sign(3-4*%i)
```

$$ans = 0.6 - 0.8i$$



Obrázek 2: Funkce signum

4.1.5 Funkce modulo

Funkce **modulo(n, m)** se používá pro výpočet zbytku při celočíselném dělení. Při volání funkce ve tvaru *modulo (n,m)* se vypočítá číslo *n - m * int (n / m),* kde *n, m* jsou reálná čísla. Funkci *modulo* můžeme také použít při ověřování dělitelnosti dvou celých čísel - číslo *n* je násobek čísla *m*, když *modulo(n, m) = 0.*

4.1.6 Funkce rat

Funkce **rat** převede desetinné číslo na zlomek. Syntaxe je **[n,d] = rat(x)**, kde *x* je desetinné číslo, *n, d* jsou celá čísla. Například převedeme číslo $\pi = 3.14159$ na zlomek:

```
--> [n,d] = rat(3.14159)
```

d = 113.

n = 355.

4.1.7 Funkce sqrt

Funkce **sqrt(x)** vypočítá druhou odmocninu čísla. Druhá odmocnina kladného reálného čísla je kladné reálné číslo. Odmocnina záporného čísla je komplexní číslo.

4.1.8 Faktoriál

Pro výpočet faktoriálu slouží funkce **gamma(x),** která je definována jako

n! = gamma(n+1).

Například 5! vypočítáme příkazem

```
--> gamma(5+1)
```

$ans = 120.$

4.2 Exponenciální a logaritmické funkce

Exponenciální funkce **exp(x)** vrací hodnotu e^x pro reálná čísla *x, e* je základ přirozených logaritmů. Tato konstanta má ve Scilabu definovanou hodnotu $e = 2.7182818$.

Exponenciální funkci můžeme použít i u komplexních čísel s vyjádřením

$$\exp(z) = e^{a+bi} = e^a \cos a + e^b i \sin b$$

Výpočet exponenciální funkce:

```
--> exp([-1 0 1 2 3])
```

$ans = [0.3678794 \quad 1 \quad 2.7182818 \quad 7.3890561 \quad 20.085537]$

```
--> exp(3+4*%i)
```

$ans = - 13.128783 - 15.200784i$

Inverzní funkce k exponenciální funkci je přirozený logaritmus *ln (x).* Ve Scilabu odpovídá přirozenému logaritmu funkce **log(x).** Kromě přirozeného logaritmu *log(x)* Scilab poskytuje funkce **log10(x)** a **log2(x),** které vypočítají logaritmy se základem 10 a 2.

```
--> log(3.5)
```

$ans = 1.252763$

```
--> log10([100 200 1000])
```

$ans = [2 \quad 2.30103 \quad 3]$

4.3 Goniometrické funkce

Mezi základní goniometrické funkce patří sinus (*sin*), kosinus (*cos*), tangens (*tan*), kotangens (*cot*). Tyto funkce jsou definovány pomocí stran a úhlů v pravoúhlém trojúhelníku. Ve Scilabu jsou předdefinovány čtyři goniometrické funkce (*sin, cos, tan, cotg*) a tři inverzní funkce arkus sinus (*asin*), arkus kosinus (*acos*) a arkus tangens (*atan*). Z exponenciálních funkcí dále vycházejí hyperbolické funkce: hyperbolický sinus (*sinh*),

hyperbolický kosinus (*cosh*) a hyperbolický tangens (*tanh*) a jejich inverze *asinh, acosh, atanh*.

Hyperbolické funkce jsou definovány:

$$\sinh(x) = \frac{e^{x} - e^{-x}}{2} \qquad \cosh(x) = \frac{e^{x} + e^{-x}}{2} \qquad \tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Všechny goniometrické funkce mají stejnou syntaxi **[y] = fce(x)**, kde *x* je vektor nebo matice.

Příklady výpočtu:

```
--> sin([0 %pi/6 %pi/4 %pi/3 %pi/2 %pi])
```

$$ans = \begin{bmatrix} 0 & 0.5 & 0.7071068 & 0.8660254 & 1 & 0 \end{bmatrix}$$

```
--> tan([0 %pi/6 %pi/4 %pi/3 %pi])
```

$$ans = \begin{bmatrix} 0 & 0.5773503 & 1 & 1.7320508 & 0 \end{bmatrix}$$

5 VIZUALIZACE

Scilab poskytuje rozsáhlé možnosti práce s grafikou, tvorbu 2D a 3D grafů a tvorbu speciálních druhů grafů.

Při vývoji Scilabu došlo k velkým změnám grafiky. Ve verzi 2.7 se používal starý grafický mód (*old graphic mode*). Se Scilabem 3.0 byla vytvořena zcela nová grafika (*new graphic mode*), objektově orientovaná, rychlejší a modernější. Ve verzi 4.0 je tento nový grafický mód nastaven implicitně a je to poslední verze Scilabu, která ještě podporuje oba grafické módy.

V nové grafice je každé grafické okno a graf v něm reprezentován hierarchickou stromovou strukturou jednotlivých objektů. Na vrcholu této struktury je **Figure** (obrázek, schéma, graf). Každý takový graf obsahuje alespoň jedno "dítě" (**Child**) typu **Axes** (osy). Každý objekt typu osa zase obsahuje sadu grafických objektů jako jsou křivky, obdélníky, oblouky nebo úsečky.

Hlavním cílem této objektově orientované grafiky je poskytnout co nejjednodušší způsob změn vlastností objektů a snadnou manipulaci s nimi. [9]

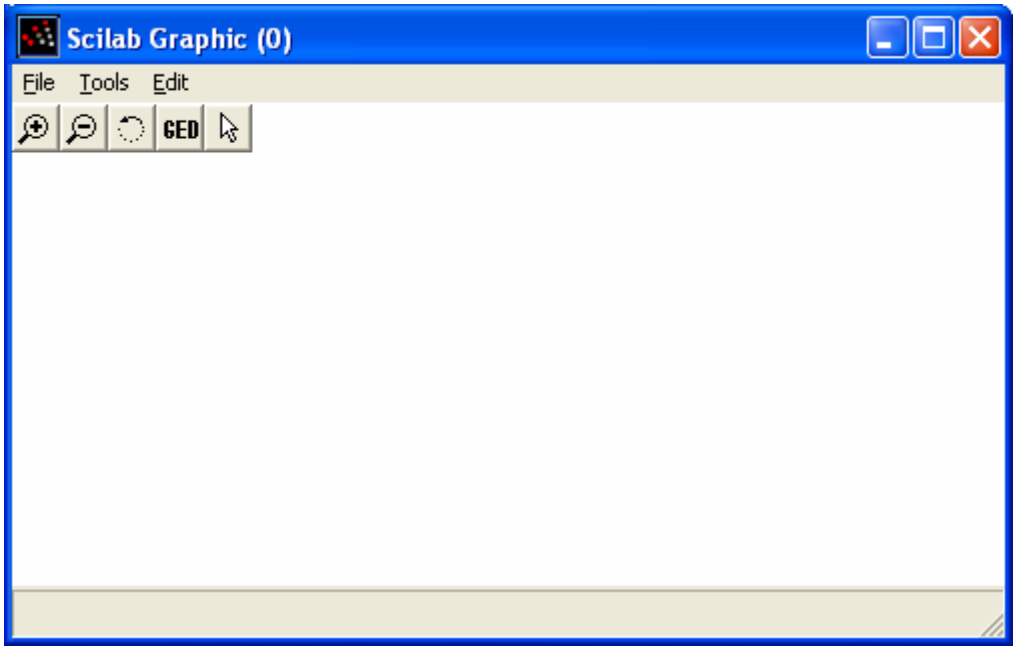
5.1 Grafické okno

Grafy se zobrazují v grafickém okně nazvaném **Scilab Graphic (x)**, jeho přednastavená velikost je 618*535 pixelů a je typu **figure**. Těchto oken může být spuštěno několik najednou, ale jen jedno může být v daném okamžiku aktivní. Grafické okno se automaticky vyvolá po zavolání funkce vytvářející graf.

V grafickém prostředí můžeme libovolně nastavovat některé parametry grafů, např. barvu grafu nebo pozadí, různou tloušťku a styl čar, nastavit mřížku, zobrazit popisky os, vytvořit název grafu atd.

Jednoduché grafické okno se vytvoří příkazem **scf()**. Pokud zadáme *scf(x),* kde *x* je číslo, vytvoří se grafické okno s tímto id číslem. Zároveň se vypíše nastavení okna, které je typu figure. Pro přístup k jednotlivým položkám slouží operátor tečka. Všechny položky u grafických oken (*figure*), os (*axes*) i objektů (*entity*) můžeme změnit, a tak dotvořit výsledné grafické zobrazení.

Prázdné grafické okno potom vypadá takto:



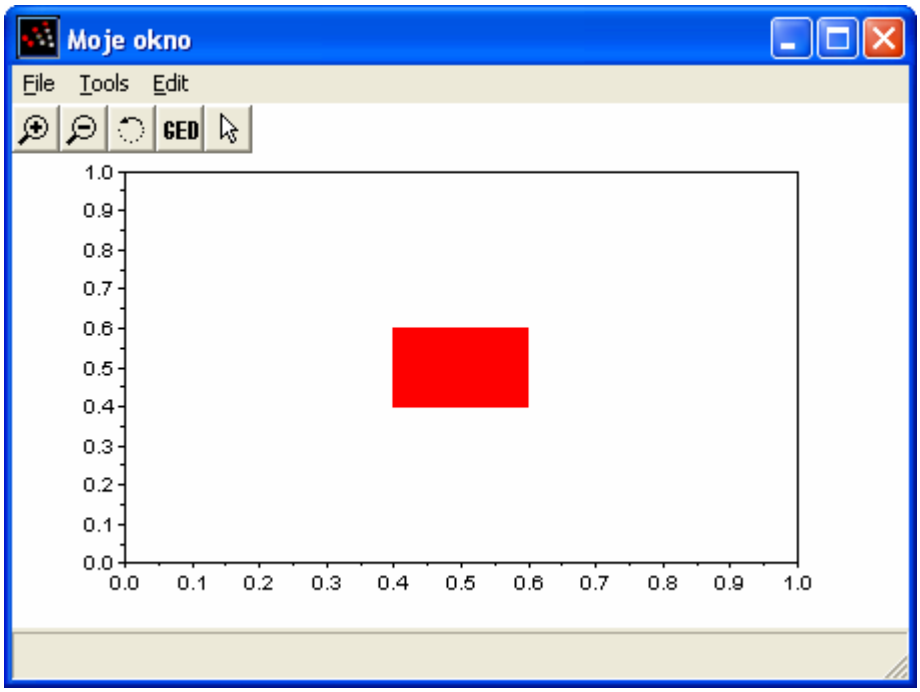
Obrázek 3: Grafické okno

Tyto tlačítka  dovolují *zoom*, *unzoom* a 2D/3D rotaci, *GED* spustí grafický editor (*Figure Propeties*) a poslední je *Entity Picker*.

V nabídce *File* nalezneme možnosti náš graf uložit, vytisknout nebo zkopírovat, můžeme také vytvořit nové grafické okno nebo okno zavřít. V *Tools* můžeme zapnout/vypnout tlačítkovou lištu a v nabídce *Edit* můžeme vylepšovat náš graf, měnit jeho nastavení a parametry.

Příklad: vytvoříme grafické okno, zobrazíme systém os a nakonec vložíme grafický objekt (obdélník).

```
--> okno=scf(1);
--> okno.figure_name="Moje okno";
--> osa=okno.children;
--> osa.axes_visible = "on";
--> osa.box="on";
--> xfrect (0.4, 0.6, 0.2, 0.2);
--> obd=gce();
--> obd.background=5;
```



Obrázek 4: Grafické okno s objektem

5.1.1 Grafické proměnné

Prvky grafického okna (handles) jsou hierarchicky uspořádané se strukturou rodič - dítě (parent - children). Pro všechny tyto prvky existuje několik funkcí, pomocí kterých lze nastavit parametry přes operátor tečka.

Funkce **scf** (*set current figure*) - vytvoří okno a vypíše jeho nastavení.

Funkce **gdf** (*get default figure*) - nastavení okna (figure).

Funkce **gcf** (*get current figure*) - nastavení aktuálního okna (figure).

Funkce **gda** (*get default axes*) - nastavení os (axes).

Funkce **gca** (*get current axes*) - nastavení nyní používaných os.

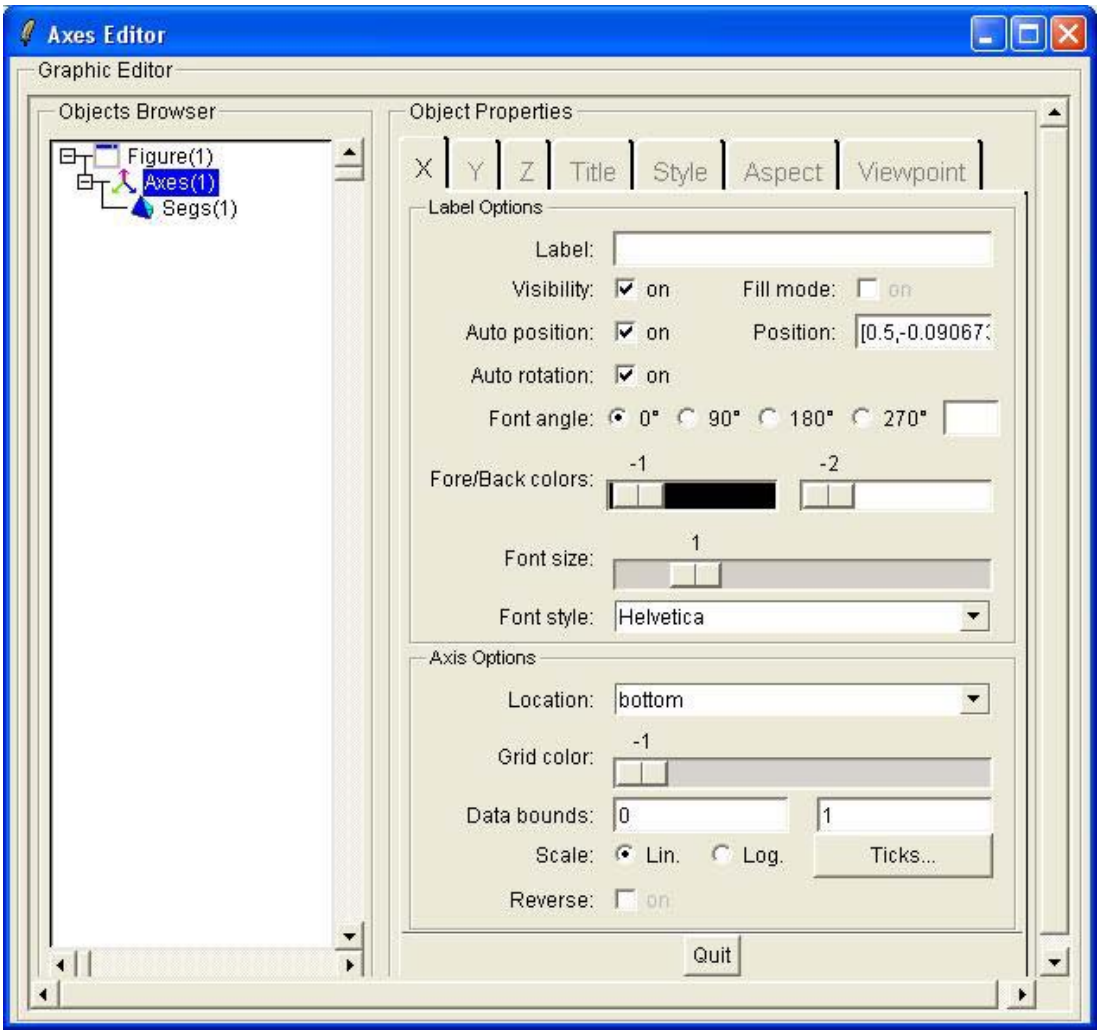
Funkce **gce** (*get current entity*) - nastavení naposledy vytvořeného objektu (entity).

5.2 Grafický editor

Pomocí grafického editoru (*GED*) provedeme jednoduché nastavení grafu. Spustíme ho kliknutím na tlačítko 

nebo v nabídce *Edit -> Figure propeties*. Otevře se grafický

editor, nebo také *Figure editor*: Jednotlivé grafické prvky jsou uspořádány ve stromové struktuře, po klepnutí na *Axes* se rozbalí nastavení pro další objekty. Vlastnosti u grafického objektu budou u různých objektů různé. Těmito objekty například mohou být: *rectangle* (obdélník), *segs* (úsečka), *polyline* (křivka), *arc* (oblouk).



Obrázek 5: Grafický editor (GED)

5.2.1 Figure

V záložce *Style* můžeme nastavit:

- *Visibility* - viditelnost grafu
- *Figure name* - název grafického okna
- různé pozice os
- *Background color* - barva pozadí grafu

5.2.2 Axes

Jsou zde 3 záložky pro nastavení os *x*, *y*, *z*. V každé máme možnosti:

- *Label* - název (popis) osy
- *Visibility* - viditelnost osy
- *Font angle* - popisek osy pod úhlem
- *Fore/Back color* - barva písma
- *Font Size* - velikost písma
- *Font Style* - typ písma
- *Location* - umístění osy (u osy *x* : *top* - nahoře, *middle* - uprostřed, *bottom* - dole, u osy *y* : *left* - vlevo, *middle* - uprostřed, *right* - vpravo)
- *Grid Color* - barva mřížky (pro hodnotu -1 je mřížka neviditelná)
- *Scale* - měřítko (*lin* - lineární, *log* - logaritmické)
- *Ticks* - otevře další okno s nastavením popisků na číselné ose

V záložce *Title* :

- *Label* - název grafu
- *Visibility* - viditelnost názvu
- *Font angle* - název pod úhlem
- *Fore/Back color* - barva písma
- *Font Size* - velikost písma
- *Font Style* - typ písma

V záložce Style:

- *Font Style* - typ písma na číselné ose
- *Font Color* - barva písma na číselné ose
- *Font Size* - velikost písma na číselné ose
- *Fore Color* - barva os (rámečku)

- *Back Color* - barva výplně grafu (rámečku)
- *Thickness* - tloušťka čáry os (rámečku)
- *Line Style* - styl čáry

V záložce *Aspect*:

- *Boxed* - uzavřený rámeček kolem os

5.2.3 Objekty

Pro jednotlivé objekty se budou možnosti nastavení mírně lišit, uvedeny budou pouze nejdůležitější:

V záložce *Style*:

- *Visibility* - viditelnost
- *Line* - styl čáry (*solid* - spojitá, *dash* - čárkovaná, *dash dot* - drobně čárkovaná, *long dash dot* -čerchovaná, ...)
- *Arrow Size* - velikost šipky
- *Mark Mode* - zobrazení značky
- *Mark Style* - druh značky (*dot* - bod, *cross* - křížek, *star* - hvězda, *circle* - kolečko)
- *Mark Size* - velikost značky
- *Mark Foreground* - barva čáry značky
- Mark Background - barva výplně značky

V záložce *Data* můžeme jednoduše editovat data grafu, vybereme v nabídce *Edit data*, otevře se nové okno *Scilab Edit Var*().

5.3 Grafické atributy

5.3.1 Barvy

Barvy jsou kódovány v **RGB** kódu (*red green blue*) vyjádřeném třemi číslicemi od 0 do 255. Tento kód umožňuje vytvořit až 16 777 216 různých barev. V základní paletě barev je definováno 32 barev. Tuto paletu můžeme zobrazit příkazem **getcolor()**, který otevře dialogové okno a kliknutím na barvu se zobrazí její podrobnosti.

Ve Scilabu existují i další palety barev, které jsou složené z různých odstínů barev : *hotcolormap* (černá, červená, žlutá a bílá), *graycolormap* (černá, šedá a bílá), *jetcolormap* (indigo, modrá, modrozelená, zelená, žlutá, oranžová, červená, hnědá) a *hsvcolormap* (barvy duhy: červená, žlutá, zelená, modrá, fialová, červená.).

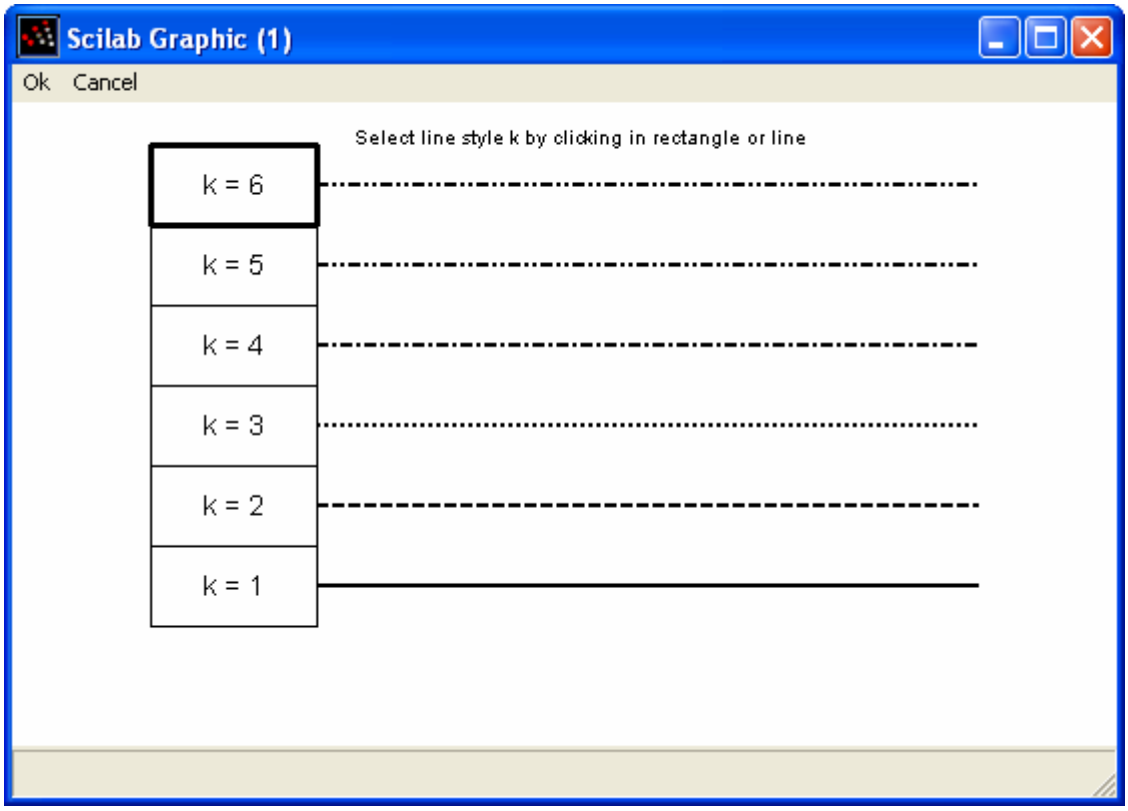
5.3.2 Čáry

Vykreslovací čáry jsou určeny třemi parametry: stylem, tloušťkou a barvou čáry.

Styl čáry je určen číslem 1 až 6, přičemž default hodnota je 0 (spojitá čára). K vykreslení všech druhů čar existuje podobný příkaz jako u barev, je to příkaz **getlinestyle**. Funkce **getlinestyle** otevře dialogové okno a zobrazí všechny druhy čar a jejich číselný kód.

Jejich anglické názvy jsou :

1. solid
2. dash
3. dash dot
4. longdash dot
5. bigdash dot
6. bigdash longdash



Obrázek 6: Druhy čar

Tloušťka čáry je vyjádřena číslem. Default hodnota je 1 (tenká čára).

5.3.3 Značky

Grafických značek ve Scilabu je celkem 15. Jejich nastavitelné parametry jsou viditelnost (*on/off*), typ značky, barva, velikost. Funkce **getmark** vykreslí všechny značky do přehledné tabulky.

Jejich anglické názvy a grafické vyjádření:

0. dot ●
1. plus +
2. cross ×
3. star ⊕
4. filled diamond ◆
5. diamond ◇
6. triangle up △
7. triangle down ▽
8. diamond plus ⌖
9. circle ○
10. asterisk ✳
11. square □

12. triangle right ▷
13. triangle left ◁
14. pentagram ☆

5.3.4 Text

Text, zobrazený v grafickém prostředí, je podmíněn třemi parametry: druh, velikost a barva písma. Pro zobrazení textu do grafického okna slouží příkaz **xstring**. Syntaxe je *xstring(x,y,str,[angle,box]),* kde *x, y* jsou souřadnice levého dolního rohu, *str* je řetězec, který chceme napsat, parametr *angle* určuje sklon písma ve stupních, *box* je rámeček kolem textu a nabývá hodnoty 0 (default, není rámeček) nebo 1 (je rámeček).

Na výběr máme z deseti druhů písma :

0. Courier
1. Symbol
2. Times
3. Times Italic
4. Times Bold
5. Times Bold Italic
6. Helvetica
7. Helvetica Italic
8. Helvetica Bold
9. Helvetica Bold Italic

Default hodnota je 6. Velikost písma je definována od 0 do 5.

Existuje i funkce **getfont**, která vypíše tabulku s druhy písma a velikostí písma. Při stisknutí písmena na klávesnici se daný znak zobrazí v tabulce.

5.4 Tvorba 2D grafů

K vytváření 2D grafů slouží ve Scilabu dvě funkce: **plot** a **plot2d**. Tyto funkce jsou téměř totožné, liší se pouze v syntaxi a zadávání parametrů. Funkce **plot** je převzatá z Matlabu i se syntaxí pro zlepšení kompatibility obou programů.

Funkce **plot2d** je původem ze Scilabu. Z této funkce vycházejí další podobné funkce : *plot2d2, plot2d3, plot2d4*.

Funkce **plot2d2** - vykresluje graf ve stupňovitých segmentech (schodkový diagram)

Funkce **plot2d3** - vykresluje křivky jako kolmice

Funkce **plot2d4** - vykresluje křivky jako šipky

Ve 2D grafice můžeme vytvářet i **sloupcové diagramy** (histogramy) nebo **koláčové grafy**. Existuje i několik funkcí, které převedou povrch ve 3D prostoru do roviny. Tyto grafy se nazývají **vrstevnicové grafy.**

5.4.1 Funkce plot

Syntaxe: **plot(x, y, < LineSpec >, < GlobalProperty >),** kde *x, y* jsou matice nebo vektory, parametry *LineSpec* a *GlobalProperty* jsou nepovinné položky.

LineSpec je volitelný parametr, který se používá pro rychlé nastavení základních stylů grafů. Píše se přímo do funkce *plot* a upravuje pouze definovanou křivku. Zapisuje se jako řetězec obsahující informace o barvě, stylu čáry nebo značce.

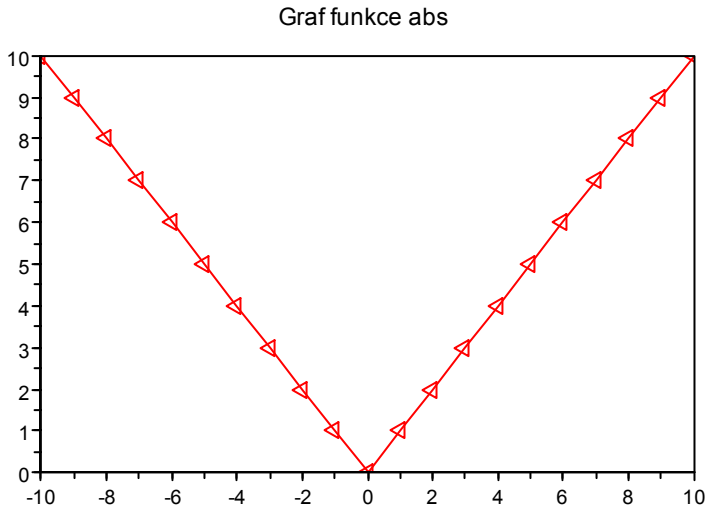
Jsou to tyto možnosti nastavení:

-	spojitá čára	**o**	značka kolečko
--	čárkovaná čára	*****	značka hvězdička
:	tečkovaná čára	**.**	značka bod
g	zelená barva	**x**	značka křížek
b	modrá barva	**s**	značka čtverec
c	modrozelená barva	**d**	značka kosočtverec
m	fialová barva	**^**	značka trojúhelník nahoru
y	žlutá barva	**v**	značka trojúhelník dolů
k	černá barva	**>**	značka trojúhelník doprava
w	bílá barva	**<**	značka trojúhelník doleva
+	značka plus	‘**pentagram**’	pentagram

Příklady:

Graf1: vytvoříme graf funkce *y = abs(x)*, barvu čáry nastavíme červenou (r), čáru spojitou (-) a značku na trojúhelník (<).

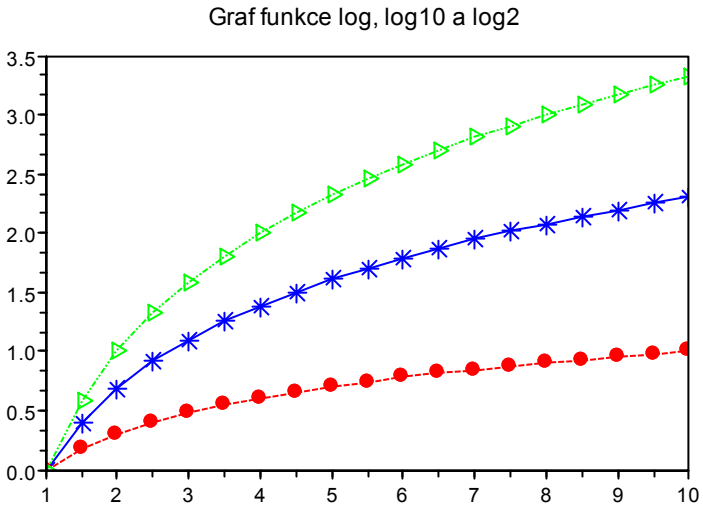
```
--> x = -10:10;
--> plot(x, abs(x),'r-<')
```



Obrázek 7: Graf funkce abs

Graf2: vytvoříme 3 grafy funkcí *log (x), log10(x) a log2(x)* do jednoho grafu. Pro každou křivku nastavíme různé parametry (barvu, značku, čáru).

```
--> x = 1:0.5:10;
--> plot(x, log(x), 'b-*', x, log10(x), 'r--.', x, log2(x),'g:>')
```



Obrázek 8: Graf funkce log, log10 a log2

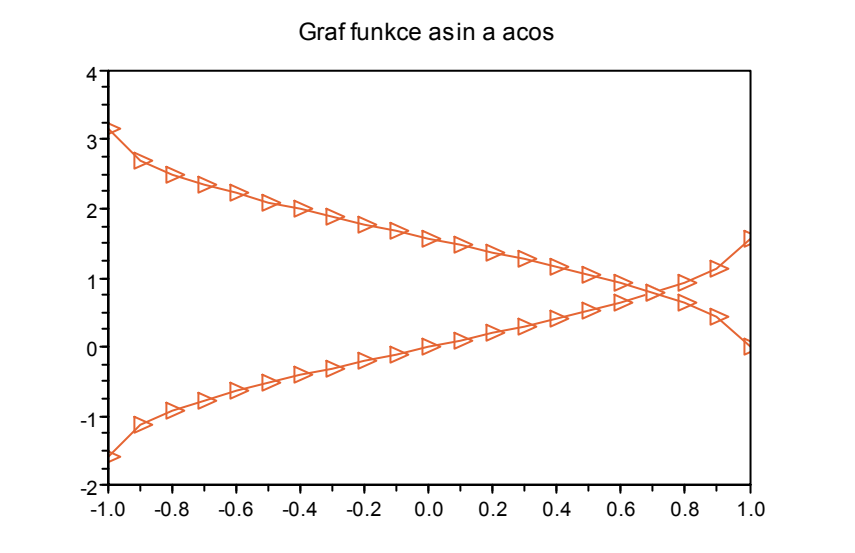
Global Property (globální parametry) slouží pro nastavení vlastností pro všechny křivky definované naráz. Tyto parametry se zapisují ve dvojici (*PropertyName, PropertyValue*). Hodnoty parametrů jsou podobné jako u *LineSpec*. Jsou to:

color, colo	barva	**markbackground**	barva výplně značky
linest, linestyle	styl čáry	**markersize**	velikost značky
marker	značka	**thickness**	tloušťka čáry
markforeground	barva značky		

Příklad:

Graf3: vytvoříme graf funkcí $y = asin(x)$ a $y = acos(x)$. Pro obě křivky nastavíme globální parametry – barvu, značku a velikost značky.

```
--> x = [-1:0.1:1];
--> plot(x, asin(x), x, acos(x), 'color', [0.9 0.4 0.2], 'marker', '>',
'markersize', 7)
```



Obrázek 9: Graf funkce asin a acos

5.4.2 Funkce plot2d

Syntaxe: **plot2d([x], y, <opt_args>),** kde *x*, *y* jsou vektory nebo matice, *opt_args* jsou nastavitelné parametry grafu.

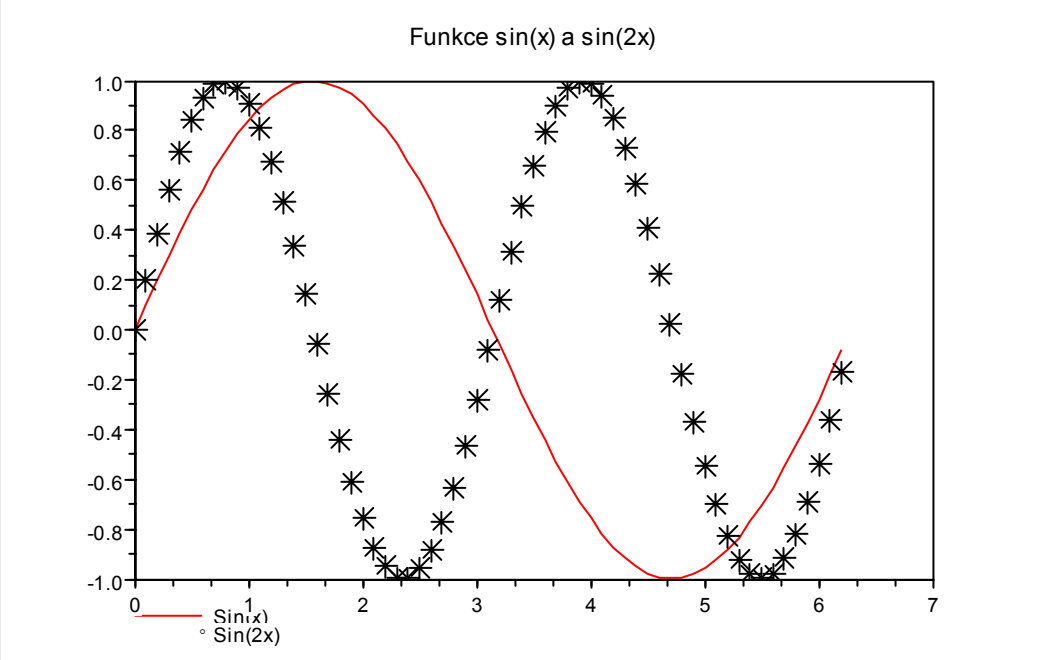
Těmito parametry jsou : *axesflag, frameflag, leg, logflag, nax, rect, style.*

Parametr **leg** zobrazí legendu pod grafem. Je to řetězec znaků ve formě **"leg1@leg2@...."**, kde *leg1*, *leg2* jsou legendy k jednotlivým křivkám nebo objektům v grafu. Tato legenda se vypisuje pod osu *x*, což není zrovna nejlepší. Pro zobrazení legendy existuje ještě funkce **legend**, která má více možností a je flexibilnější. Parametr **axesflag** se používá pro různé druhy nastavení os. Toto nastavení lze provést i v grafickém editoru (GED). Parametr **rect** definuje obdélník s limitními hranicemi pro graf. Hodnota parametru *rect* je vektor se 4 položkami : *[xmin, ymin, xmax, ymax]*. Parametr **logflag** definuje typ měřítka pro každou osu. Je charakterizován řetězcem *"xyz"*, kde *x, y, z* je buď "*n*" (normální měřítko) nebo "*l*" (logaritmické měřítko). Parametr **nax** definuje typ dílkování stupnice každé osy ve formě vektoru čísel *[sgx, gx, sgy, gy]*. Default hodnota je [1, 11, 1, 11]. *Gx* (*gy*) je počet dílků stupnice na ose x (y), *sgx* (*sgy*) je počet dalších dílků mezi dvěma dílky stupnice.

Možnost **style** definuje styl vykreslení čáry grafu. Pokud je kladné číslo, čára má barvu z palety podle čísla, pokud je záporné číslo nebo nula, místo čáry je vykreslena značka.

Příklad: vytvoříme graf funkcí *y = sin(x)* a *y = sin(2x)*. Pro obě křivky zobrazíme legendu (parametr *leg*) a styl čáry (parametr *style*).

```
--> x=[0:0.1:2*%pi]';
--> plot2d(x,[sin(x) sin(2*x)], leg="Sin(x)@Sin(2x)", style = [5,-10])
```



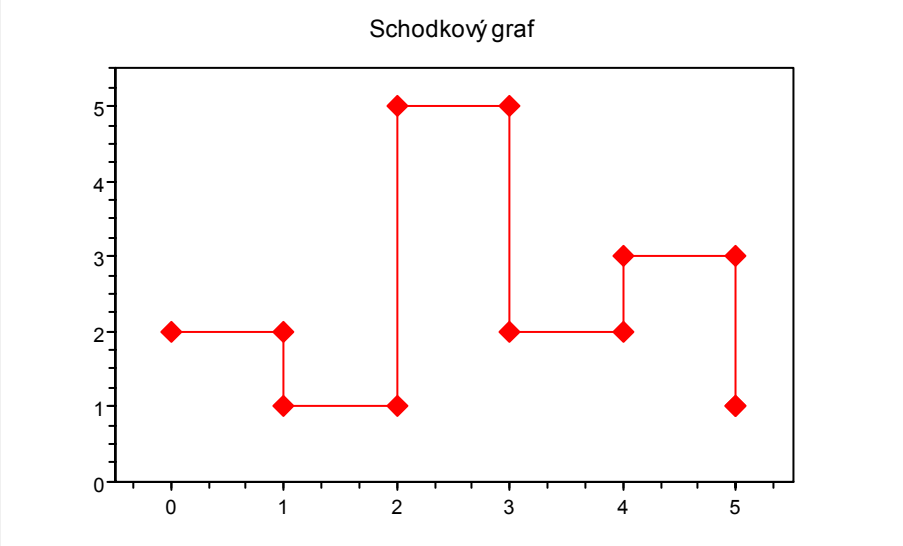
Obrázek 10: Funkce sin(x) a sin(2x)

Funkce **plot2d2**, **plot2d3** a **plot2d4** mají stejnou syntaxi i parametry jako funkce **plot2d**. Funkce **plot2d2** vykreslí schodový graf – graf ve stupňovitých segmentech. Funkce **plot2d3** vykreslí kolmicový graf – jednotlivé body grafu jako svislé čáry (kolmice k ose x). Funkce **plot2d4** vykreslí body grafu jako šipky.

Příklady:

Graf1: Schodkový graf, kde zobrazíme vybranou část grafu (parametr rect) a potom nastavíme značku.

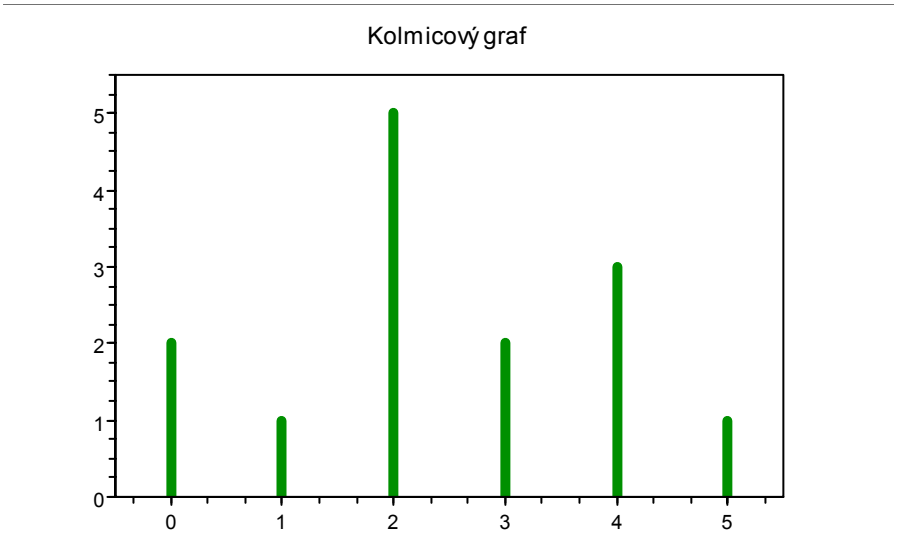
```
--> y = [2, 1, 5, 2 ,3, 1];
--> plot2d2 ([0:5],y, rect=[-0.5, 0, 5.5, 5.5], style = 5);
--> graf = gce();
--> st= graf.children;
--> st.mark_style = 4 ;
--> st.mark_foreground = 5;
```



Obrázek 11: Schodkový graf

Graf2: Kolmicový graf, použijeme stejný příklad jako u *grafu 1*, jen změníme některé parametry.

```
--> y = [2, 1, 5, 2 ,3, 1];
--> plot2d3 ([0:5],y, rect=[-0.5, 0, 5.5, 5.5], style = 13);
```



Obrázek 12: Kolmicový graf

5.4.3 Speciální 2D grafy

Mezi speciální grafy zařadíme histogram, koláčový graf a vrstevnicový graf.

Sloupcový diagram neboli **histogram** se používá ke znázornění četností dat v matematické statistice. Hodnoty znaku jsou sdruženy do intervalů *x*. Tyto intervaly tvoří základny sloupků, odpovídající četnosti pak udávají jejich výšky.

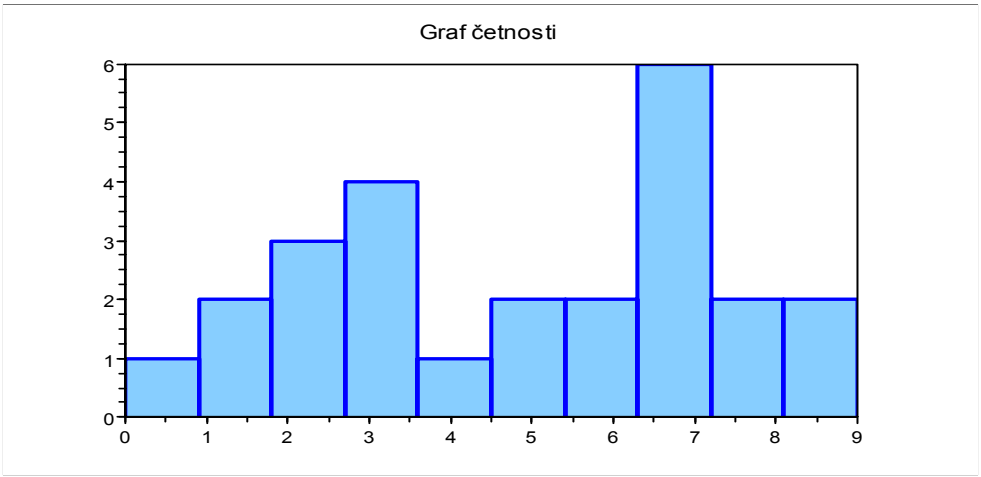
Syntaxe: **histplot(x, data, <opt_args>** kde *x* udává počet intervalů, *data* je soubor (matice) dat, například celých čísel. Položka *opt_args* jsou volitelné parametry a jejich zápis je stejný jako u funkce plot2d. Graf se vykreslí v normalizovaném tvaru. Pokud nechceme normalizovaný graf, stačí do parametrů grafu zapsat *normalize = %f (false).* Default hodnota je *%t (true).*

Příklad: Vygenerujeme 25 celých čísel v intervalu 0 až 9 a určíme jejich četnost výskytu. pomocí grafu. Graf vykreslíme v nenormalizovaném tvaru.

```
--> d=int(rand(5,5)*10);
```

$$d = \begin{matrix} 1 & 3 & 7 & 3 & 5 \\ 7 & 5 & 7 & 9 & 1 \\ 6 & 3 & 2 & 2 & 2 \\ 0 & 4 & 6 & 7 & 7 \\ 8 & 8 & 7 & 3 & 9 \end{matrix}$$

```
--> histplot(10,d, normalization = %f)
```

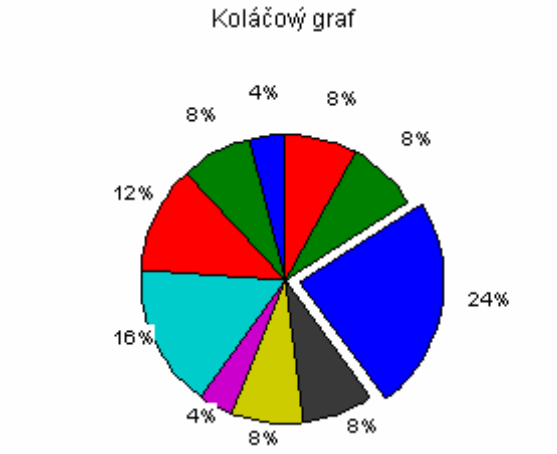


Obrázek 13: Histogram – graf četnosti

Koláčový graf má syntaxi **pie(x, [sp], [txt])** kde *x* je vektor se vstupními daty. *Sp* a *txt* je volitelná položka. *Sp* umožňuje vyřezání a posunutí jednoho nebo více klínů z grafu směrem od středu. Zapisuje se v podobě vektoru nul (neposunutý) a jedniček (posunutý), který musí mít stejnou velikost jako vstupní data. *Txt* je vektor taky o stejné velikosti jako vstupní data a dovoluje napsat text pro každou položku grafu. Popisek jednotlivých dílků se automaticky vytvoří v procentech. Vlastní popisek dílků můžeme zapsat přímo do příkazového řádku v položce *txt*, nebo pomocí grafického editoru (GED). Navíc některé dílky můžeme posunout ven (například dílky s největší četností).

Příklad: použijeme data z předchozího příkladu a zobrazíme je do koláčového grafu. Dílek s nejvyšší četností posuneme ven.

```
--> x=[1 2 3 4 1 2 2 6 2 2];
--> sp = [0 0 0 0 0 0 0 1 0 0];
--> pie(x,sp)
```



Obrázek 14: Koláčový graf

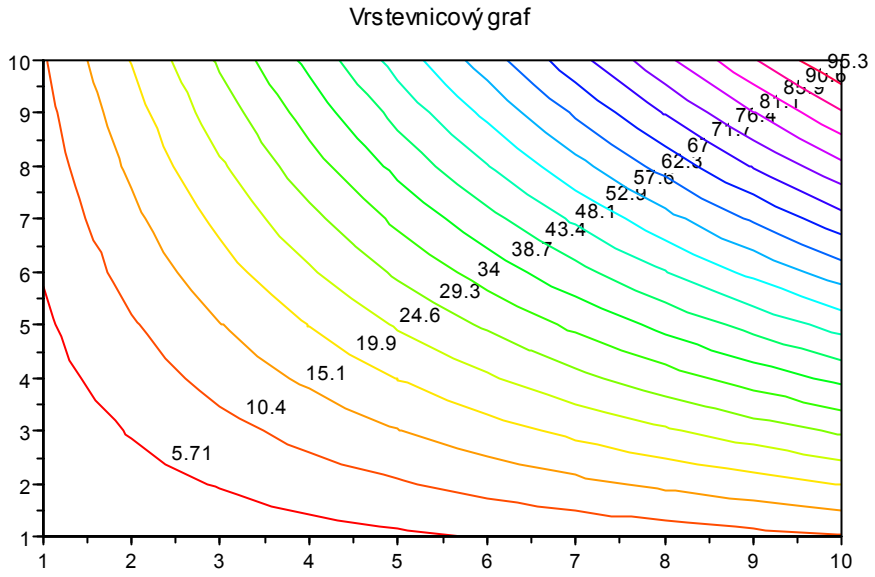
Vrstevnicové grafy vykreslí hladiny povrchu *z=f(x,y)* do 2D grafu.

Syntaxe: **contour2d(x, y, z, nz,< opt_args>)**

Hodnoty *f(x,y)* jsou dány maticí *z* v mřížce bodů definovaných pomocí vektorů *x* a *y*. *Nz* vyjadřuje počet hladin (vrstevnic). Téměř stejná je i funkce **contourf**, která jednotlivé plochy barevně vyplní.

Příklad: do vrstevnicového grafu vykreslíme plochu $z = x \cdot y$.

```
--> f=gcf();
--> f.color_map=hsvcolormap(20);
--> x = 1:10; y = 1:10;
--> z = x'*y;
--> contour2d(x,y,z,20);
```



Obrázek 15: Vrstevnicový graf

5.4.4 Legenda

Pro zobrazení legendy do okna grafu slouží příkaz **legend**.

Syntaxe: **legend (string1, string2, ... [,pos] [,boxed]),** kde parametry

string1, string2, **...** jsou řetězce, názvy křivek v grafu, které chceme zapsat do legendy

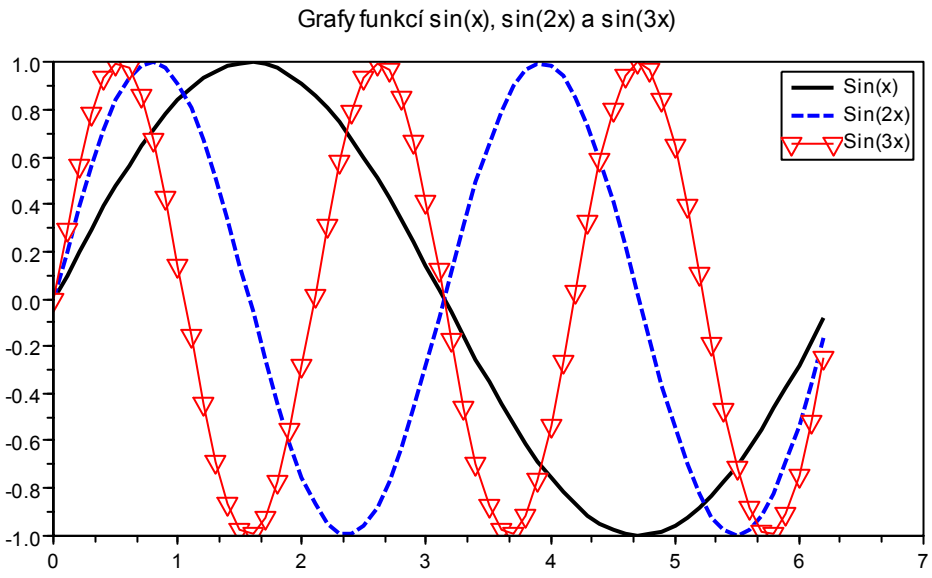
Pos specifikuje, kam umístit legendu. Jako první možnost zapíšeme souřadnice umístění horního levého rohu legendy (vektor [x,y]). Jako druhou možnost zvolíme přednastavené hodnoty. Tyto hodnoty jsou:

1. horní pravý roh (default)
2. horní levý roh
3. dolní levý roh
4. dolní pravý roh
5. umístění legendy kliknutím myši na místo

boxed - zobrazení rámečku kolem legendy (default %t)

Příklad: nakreslíme 3 grafy: *sin(x), sin(2x) a sin(3x)* a zobrazíme pro tyto křivky legendu v pravém horním rohu (hodnota 1).

```
--> x=[0:0.1:2*%pi]';
--> plot2d(x,[sin(x) sin(2*x) sin(3*x)]);
--> legend('Sin(x)', 'Sin(2x)', 'Sin(3x)', 1);
```



Obrázek 16: Graf s legendou

5.5 Tvorba 3D grafů

Scilab umožňuje jednoduchou práci s prostorovou grafikou. Trojrozměrné grafy jsou vykreslovány do prostorových souřadnic **X, Y, Z**. Grafy (3D plochy) jsou definovány rovnicí **z = f (x, y)**.

Pro vykreslení jednoduchých 3D ploch existuje funkce **plot3d**. U této funkce můžeme nastavit velké množství parametrů jako jsou barvy, různé zobrazení grafu a os nebo mřížky, zobrazení legendy nebo nastavení pozorovacího bodu.

Z funkce *plot3d* vycházejí další tři funkce *plot3d1, plot3d2 a plot3d3*.

Funkce **plot3d1** je téměř totožná s funkcí *plot3d* s tím rozdílem, že graf vykreslí v barevné paletě. Funkce **plot3d2** dokáže vykreslit složité plochy. Funkce **plot3d3** je zase stejná s funkcí *plot3d2*, ale výsledný graf vykreslí v síťovém modelu.

Pro vyjádření parametrických grafů se používá funkce **param3d**. Touto funkcí můžeme vykreslit třeba šroubovici.

Stejně jako pro 2D grafiku, jsou i pro 3D grafiku přebrány funkce z programu Matlab pro zlepšení kompatibility obou programů a pro zjednodušení práce se Scilabem. Jsou to funkce **surf** a **mesh**.

V 3D zobrazení můžeme také vytvářet histogramy (funkce **hist3d**) nebo vrstevnicové grafy (funkce **contour**).

5.5.1 Funkce plot3d a plot3d1

Funkce **plot3d** vykreslí plochu *z = f (x, y)* v 3D zobrazení, kde souřadnicový systém má definované tři osy *x, y* a *z*.

Syntaxe: **plot3d(x, y, z, < opt_args >)** nebo **plot3d(x, y, z, [theta, alpha, leg, flag, ebox]).**

kde *x, y* jsou řádkové vektory o rozměru *n1* a *n2* (osa x a osa y). Tyto vektory musí být monotónní. Z je matice o velikosti (*n1,n2*), *z* je funkcí *x, y : z = f(x,y)* .O*pt_args*:jsou různá nastavení grafu, kde jednotlivé položky můžou být *theta, alpha, leg, flag, ebox.*

Parametry **theta, alpha** vyjadřují hodnoty ve stupních udávající pozorovací bod

Parametr **leg** je řetězec definující název os v grafu oddělený znakem @, například *"X@Y@Z"*.

Parametr **flag** je vektor obsahující tři položky : *flag = [mode,type,box]*

- **mode** vyjadřuje číslo barvy pro vybarvení plochy. Pokud je toto číslo *mode* > 0, plocha se vykreslí v zadané barvě s viditelnou mřížkou. Pokud je *mode* = 0, zobrazí

se jen mřížka (nebo také síť) plochy. Pro hodnotu *mode* < 0 se vykreslí plocha v barvě ale bez mřížky.

- **type** - nastavení měřítka

  type = 0 souřadnice jsou zobrazeny podle současného nastavení

  type = 1 automatická změna měřítka souřadnic, rozmezí souřadnic vychází z položky ebox

  type = 2 automatická změna měřítka souřadnic, měřítko vychází z daných dat

  type = 3 isometrické nastavení os (všechny tři osy ve stejném měřítku), vychází z položky ebox

  type = 4 isometrické nastavení os, vychází z daných dat

  type = 5 roztažené isometrické nastavení, vychází z položky ebox

  type = 6 roztažené isometrické nastavení, vychází z daných dat

- **box** - zobrazení rámečku kolem grafu

  box = 0 žádná položka kolem grafu se nezobrazí (osy, legenda, ani rámeček)

  box = 1 stejné jako box = 0

  box = 2 zobrazení pouze skrytých os

  box = 3 zobrazení os, rámečku a legendy, ale ne stupnice

  box = 4 zobrazení všech prvků grafu

Parametr **ebox** nastaví hranice pro vykreslení grafu, je vyjádřen vektorem *[xmin, xmax, ymin, ymax, zmin, zmax]*. Tento parametr se používá společně s položkou *type* (type = 1, 3 nebo 5) v parametru *flag*. Pokud *flag* chybí, *ebox* se ignoruje.

Pomocí grafického editoru (GED) můžeme snadno nastavit některé parametry plochy. Jsou to například:

color mode - barevný mód (stejné jako položka mode u parametru flag)

foreground - barva mřížky

hidden color - barva spodní části plochy

thickness - tloušťka mřížky

color flag - pro 1 je plocha vybarvena podle barevné palety, pro 0 zůstává jedna barva

mark mode - zapne/vypne zobrazení značky, další položky jsou různá nastavení pro značky

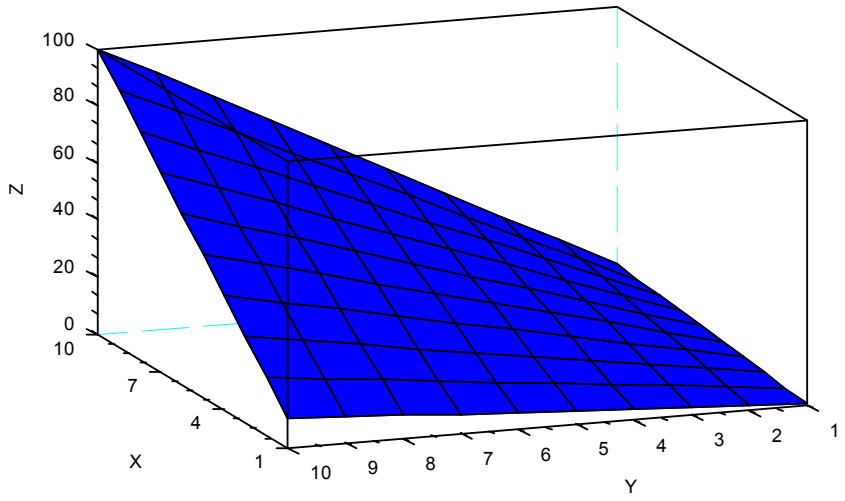
Funkce **plot3d1** se od **plot3d** liší tím, že výslednou plochu vykreslí v barvách definované barevné palety. Syntaxe i parametry jsou stejné jako u *plot3d*.

Pokud nezvolíme barevnou paletu, program použije pro vykreslení plochy default paletu barev.

Příklady:

Graf1: vykreslíme plochu z = x'*x a nastavíme vhodné hodnoty pozorovacího bodu *alpha* a *theta*.

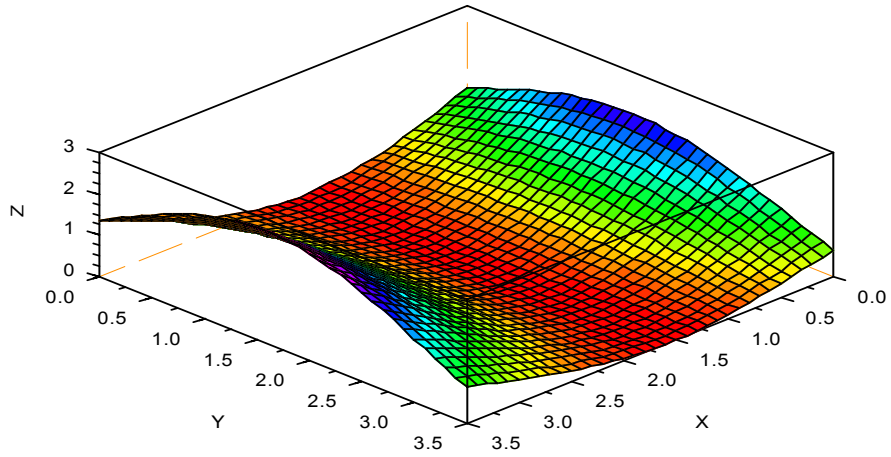
```
--> x = 1:10;
--> z = x'*x;
--> plot3d(x, x, z, alpha=12, theta=160)
```



Obrázek 17: Funkce plot3d, nastavení pozorovacího bodu

Graf2: definujme plochu *z = (1-sin(x).(1+sin(x)),* kterou vykreslíme s mřížkou a v barevné paletě *hvscolormap*. K tomu použijeme funkci *plot3d1*.

```
--> x = 0:0.1:3.5;
--> f = gcf();
--> f.color_map = hsvcolormap(32);
--> plot3d1(x, x, (1-sin(x))'*(1+sin(x)))
```



Obrázek 18: Funkce plot3d1

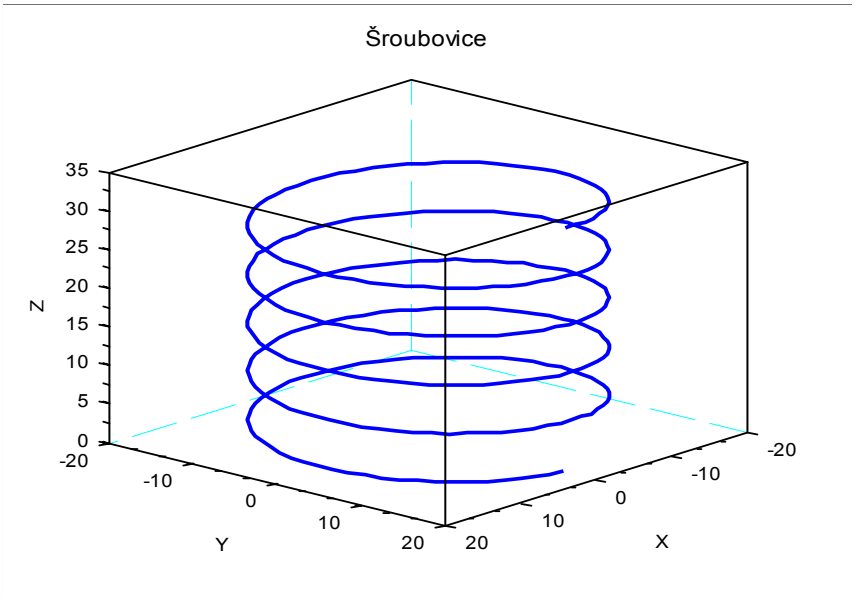
5.5.2 Funkce param3d

Funkce **param3d** se používá pro zobrazení křivek v prostoru. Pro násobný graf existuje funkce **param3d1**.

Syntaxe: **param3d(x,y,z,[theta,alpha,leg,flag,ebox]),** kde všechny parametry jsou stejné jako u funkce plot3d2.

Příklad: šroubovice

```
--> t= 0:0.1:10*%pi;
--> param3d(16*sin(t),16*cos(t),t,42,68);
```



Obrázek 19: Šroubovice

5.5.3 Funkce plot3d2 a plot3d3

Funkce **plot3d2** a **plot3d3** se používá pro vykreslení složitějších ploch. Tyto plochy jsou definovány jako *Z = f(X, Y),* kde *X,Y* a *Z* jsou matice a popisují danou plochu. Plocha je potom poskládaná z obdélníků. Funkce *plot3d3* vykreslí graf v síťovém modelu a má stejnou syntaxi jako funkce *plot3d2*

Hlavní rozdíl mezi funkcemi *plot3d* a *plot3d2* je ten, že u funkce *plot3d* musí být proměnné *X* a *Y* monotónní vektor, u funkce *plot3d2* jsou *X* a *Y* matice. To nám umožňuje vykreslit v podstatě jakoukoli plochu. Matice *X*, *Y* a *Z* musí mít stejnou velikost (rozměr). Syntaxe i parametry jsou stejné jako u funkce *plot3d*:

plot3d2(X,Y,Z [vect, theta, alpha, leg, flag, ebox]) a

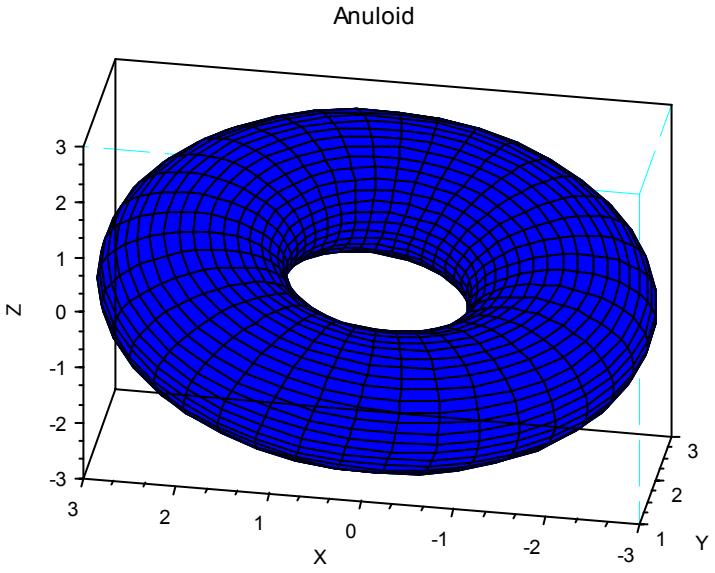
plot3d3(X,Y,Z [vect, theta, alpha, leg, flag, ebox])

Příklady:

Graf1: anuloid

```
--> u = linspace(0,2*%pi,40); v = u;
--> x = (sin(u)+2)'*cos(v);
--> y = (cos(u)+2)'*ones(v);
--> z = (sin(u)+2)'*sin(v);
```

```
--> plot3d2(x, y, z, alpha=129, theta=80);
```

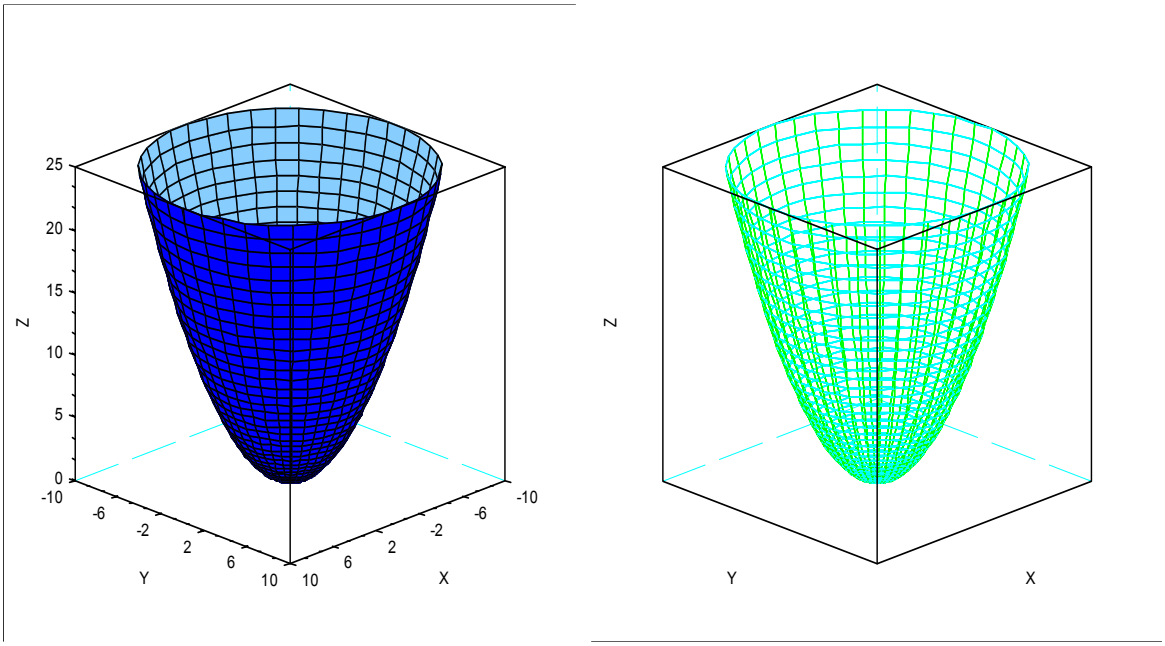


Obrázek 20: Anuloid

Graf2: Rotační paraboloid zobrazený normálně a v síťovém modelu.

```
--> u = linspace(0,2*%pi,40);v = linspace(0,10,40);
--> x = v'*cos(u);
--> y = v'*sin(u);
--> z = (v^2./4)'*ones(u);
--> plot3d2(x,y,z, alpha=65, theta=45);

Síťový model:
--> plot3d3(x,y,z, alpha=65, theta=45);
```



Obrázek 21: Rotační paraboloid v normálním a síťovém modelu

5.5.4 Funkce surf a mesh

Syntaxe: **surf(Z,< GlobalProperty >)** nebo **surf(X,Y,Z,< GlobalProperty >),**

mesh(Z) nebo **mesh(X,Y,Z),**

kde *Z* je matice reálných čísel o rozměru $m \times n$ definující plochu. Tento parametr nesmí být vynechán. *X, Y* jsou reálné vektory nebo matice o stejné velikosti

Global Property je volitelný argument. Je zapisován ve dvojici **(**PropertyName, PropertyValue**)**, který udává globální nastavení pro všechny nově vytvořené křivky. Používá se u funkcí **plot** a **surf**. Podrobnější popis je k nalezení u funkce *plot*.

Funkce **surf** vykreslí barevnou plochu v souřadnicích definovaných vektory (nebo maticemi) **X** a **Y**. Pokud nejsou **X** a **Y** stanoveny, jsou souřadnice pro vykreslení grafu určeny podle matice **Z**.

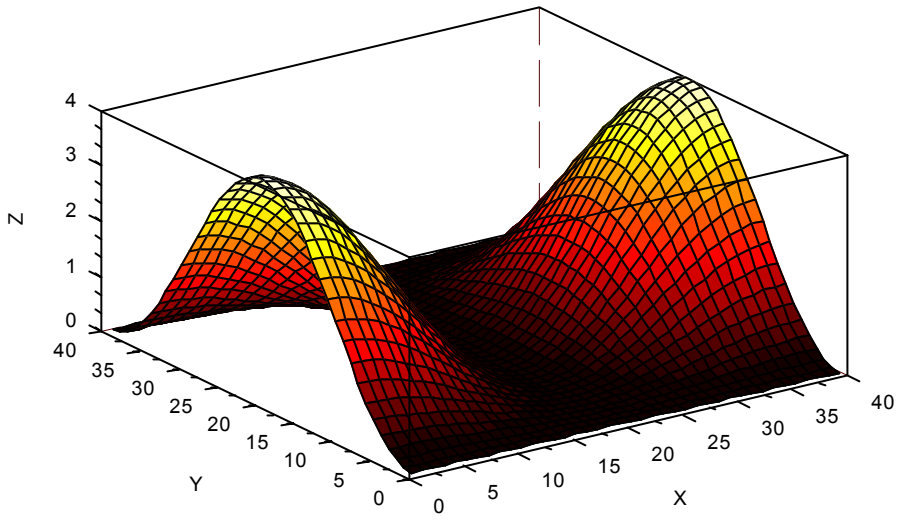
Jestliže specifikujeme pouze matici **Z**, pak **surf(Z)** vykreslí matici **Z**, síť souřadnic **X** je definována jako **1 : size(Z, 2)** a **Y** jako **1 : size(Z, 1)**. Pokud jsou dány všechny souřadnice **X**, **Y** a **Z**, pak **Z** musí být matice s rozměrem **Z = [m x n]**, **X** a **Y** jsou vektory : **X** je vektor o délce **n** a **Y** je vektor o délce **m**, nebo matice : rozměr matice **X** a **Y** musí být stejný jako rozměr **Z**.

Funkce **mesh** vykreslí graf bez barev jen v síťovém modelu. Má stejnou syntaxi i parametry jako funkce **surf**. Funkce **mesh** vychází z funkce **surf**, kde nastavení barvy grafu **color mode = 0** (bílá barva).

Obě tyto funkce byly vytvořeny pro lepší kompatibilitu s programem Matlab (podobně jako funkce *plot*).

Příklad: vykreslíme plochu $z = (1-\cos(u))(1+\cos(u))$. Souřadnice X a Y se vypočítají z matice Z. Nakonec ještě nastavíme paletu barev *hotcolormap*.

```
--> u = linspace(0, 2*%pi, 40);
--> z=(1-cos(u))'*(1+cos(u));
--> surf(z);
--> gr = gcf();
--> gr.color_map = hotcolormap(32);
```



Obrázek 22: Funkce surf

5.5.5 Speciální 3D grafy

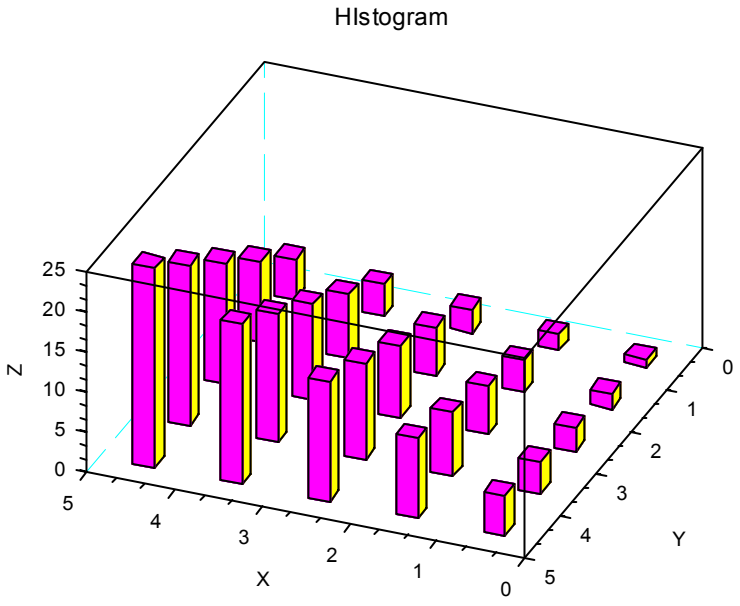
Mezi speciální 3D grafy zařadíme 3D histogram a vrstevnicový graf.

Pro zobrazení **histogramu** ve 3D grafice existuje funkce **hist3d**.

Syntaxe: **hist3d (f, [theta, alpha, leg, flag, ebox]),** kde f je matice o velikosti (m,n) definující data histogramu, *theta, alpha, leg, flag, ebox* jsou parametry stejné jako u *plot3d.*

Příklad: vytvoříme histogram matice z o velikosti 5×5 s hodnotami od 1 do 25.

```
--> x = 1:5;
--> z = x'*x;
--> gr = gcf();
--> gr = gcf();
--> hist3d(z);
--> gra = gr.children;
--> gra.rotation_angles = [10,112];
```



Obrázek 23: 3D histogram

Funkce **contour** vykreslí vrstevnicové křivky plochy *z=f(x,y)*. Tyto křivky se zobrazí na 3D graf. Syntaxe: **contour(x,y,z,nz,[theta,alpha,leg,flag,ebox]),** kde *x, y* jsou vektory o velikosti *n1* a n2, *z* je matice o rozměru (*n1, n2*), hodnoty vycházejí z funkce definující plochu, *nz* je počet vrstevnic, *theta, alpha, leg, flag, ebox* jsou parametry stejné jako u *plot3d*.

Pouze u parametru *flag = [mode, type, box]* má položka *mode* zvláštní význam:

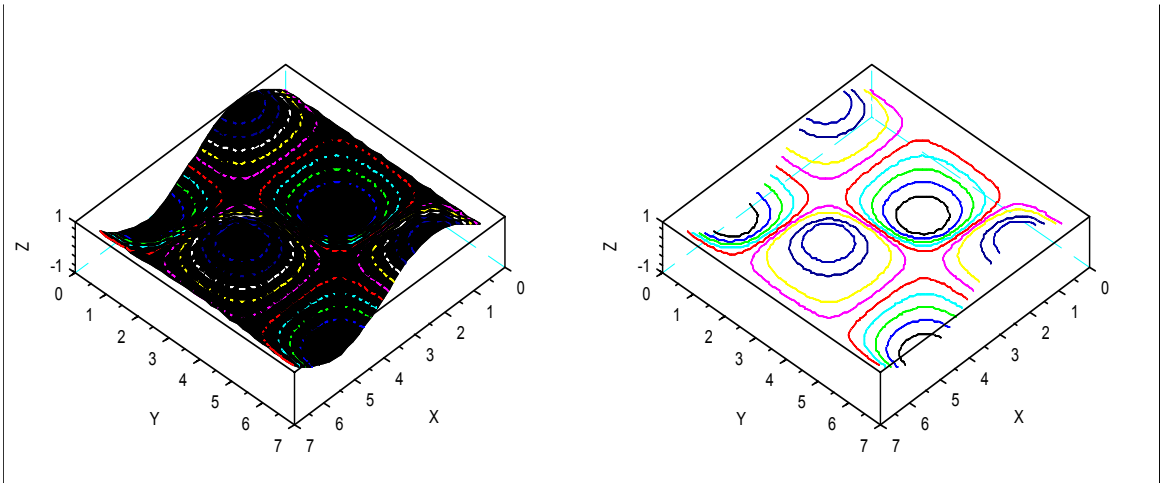
mode = 0 vrstevnice jsou vykresleny přímo na plochu definovanou jako (x, y, z)

mode = 1 vrstevnice jsou vykresleny jako 3D graf

mode = 2 vrstevnice jsou vykresleny jako 2D graf

Příklad: vytvoříme plochu $z = sin(t)cos(t)$ a vykreslíme na danou plochu vrstevnice. Barvu grafu nastavíme na černou, aby barevné vrstevnice lépe vynikly a potom vypneme zobrazení plochy.

```
--> x = 1:5;
--> z = x'*x;
--> gr = gcf();
--> gr = gcf();
--> hist3d(z);
```



Obrázek 24: Vrstevnicové 3D grafy

6 PRÁCE SE SOUBORY

Vstup a výstup dat je důležitou součástí funkcí a skriptů ve Scilabu, zejména když zpracováváme větší množství dat. Data se ukládají do souborů, binárních nebo textových. Práce s daty a se soubory je jednoduchá. Většina příkazů vychází z programovacího jazyka C nebo z Fortranu, ale jsou zde i původní funkce.

Hlavní okno je standardní vstupní a výstupní zařízení. Tato zařízení (někdy nazývány logické jednotky) se používají pro uložení nebo načtení informací z programu. Jednotlivá zařízení mají identifikační číslo. Například okno Scilabu, když se používá pro vstup, má id 5, pro výstup id 6. Tyto hodnoty jsou uloženy ve speciální proměnné *%io*.

Soubory mohou být textové s příponou *txt*, binární bez přípony (nebo s libovolnou příponou, např. dat). Delší zdrojové kódy, skripty nebo funkce si můžeme napsat v editoru *Scipad*, kde je lze uložit s příponou *sce* nebo *sci*. [8]

6.1 Binární a textové soubory

Ukládání dat do souboru a načítání dat ze souboru může být někdy velmi užitečné, zvláště když pracujeme s velkým množstvím dat. Scilab například dokáže uložit všechny proměnné, které jsme si při práci vytvořili. Tato data ukládá jako binární soubory nebo textové soubory.

Pro práci s binárními soubory se používají funkce **save** a **load**. Funkce **save** uloží data do souboru, funkce **load** načte data ze souboru. Pro vytváření textových souborů slouží funkce **file**. Tato funkce má složitou syntaxi a umožňuje vytváření nových souborů, otevírání souborů, zápis a čtení dat ze souboru nebo smazání souboru.

6.1.1 Diary files

Scilab umožňuje nejenom uložit proměnné do souboru, ale také uložit celou posloupnost příkazů jako text. K tomu existuje funkce **diary**.

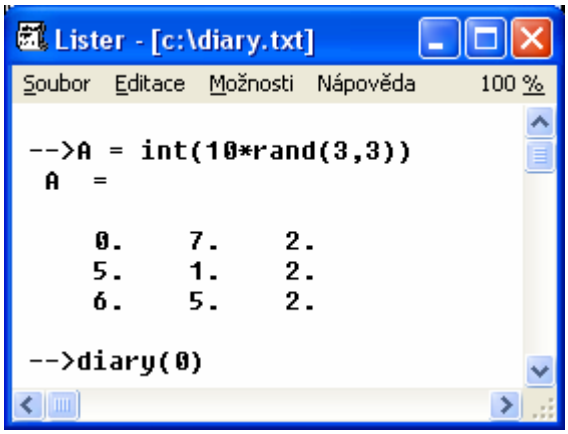
Na začátku zápisu se zavolá funkce *diary(soubor),* kde *soubor* je jméno souboru, do kterého chceme zapisovat. Po zavolání této funkce se všechno, co napíšeme do

příkazového řádku, zapíše a uloží do souboru. Tento zápis ukončíme příkazem *diary(0)*. [8]

Například vytvoříme matici, kterou si uložíme do souboru *diary.txt*.

```
--> diary('c:\diary.txt')
--> A = int(10*rand(3,3))
--> diary(0)
```

A takto vypadá výsledek v prohlížeči:



Obrázek 25: Diary files

6.2 Funkce jazyka C

Scilab obsahuje množství funkcí, které jsou podobné funkcím vyššího programovacího jazyka C (nebo C++). Funkce **printf** a **scanf** pracují přímo v okně Scilabu, zatímco funkce **fprintf** a **fscanf** pracují se soubory. Funkce pro práci s řetězci jsou **sprintf** a **sscanf**.
Ve Scilabu existují novější funkce pro práci se soubory, které vycházejí z funkcí jazyka C. Jsou to funkce **mopen** (otevře nebo vytvoří soubor na disku), **mclose** (zavře soubor) a **meof** (zjištění konce souboru). Jako dodatek k těmto třem funkcím byly přidány nové funkce **mprintf, mfprintf, msprintf, mscanf, mfscanf, msscanf**, které jsou totožné se staršími funkcemi **printf, fprintf, sprintf, scanf, fscanf, sscanf**. [8]

Pro zobrazení proměnné jako čísla se používá konverze – sekvenční znaky, stejné jako v jazyce C (např. i(*integer*), f(*float*), c(*char*), s(*string*),...). Sekvence začíná znakem procento % a končí znakem určující typ argumentu, který se bude tisknout. Dalšími znaky pro formátování jsou escape sekvence. Nejčastěji používanými znaky jsou *\n* (nový řádek) a *\t* (tabulátor).

6.3 Funkce input a disp

Funkce **input** se používá pro vstup z hlavního okna Scilabu.

Syntaxe : **[x] = input (message)**

Při volání funkce je *x* proměnná, do které se uloží hodnota zadaná z klávesnice, *message* je text (nápověda, výzva uživateli, co se má zadat). Po zavolání funkce *input* program čeká na zadání dat z klávesnice a stisknutí klávesy *enter*. Funkce *input* je užitečná, když chceme zadávat hodnoty proměnných při spuštění nějakého skriptu nebo funkce.

Funkce **disp** se používá pro výstup na obrazovku.

Syntaxe: **disp (x1, [x2,...xn])**

Hodnoty *x1, x2,..., xn* mohou být proměnné, konstanty nebo řetězce. Jako argument funkce *disp* musí být alespoň jedna hodnota (nebo více). Pokud je argumentů více, jsou zobrazeny od konce každý na jednom řádku.

7 PROGRAMOVÁNÍ

Důležitým prvkem ve Scilabu je možnost vytvářet vlastní funkce nebo skripty. Ty jsou velmi užitečné, když potřebujeme opakovat nějakou posloupnost příkazů. Součástí Scilabu je textový editor **Scipad**, který umožňuje zapisovat a ukládat skripty nebo funkce ve formátu *.sce* nebo *.sci*. Spustíme ho v nabídce *Editor* nebo příkazem *scipad*. Scipad slouží jako textový editor a zároveň debugger (ladicí program). Jednotlivé příkazy se zapisují do řádků. Syntaxe příkazů a komentářů je pro větší přehlednost barevně odlišená.

Scilab také umí pracovat s programovacími nástroji jako jsou podmínky a cykly.

Program ve Scilabu je uspořádaná posloupnost **instrukcí**. Instrukce jsou příkazy, které říkají, co se má udělat. Instrukce mohou být klíčová slova, názvy funkcí, identifikátory, oddělovače a komentáře. Oddělují čárkou, středníkem nebo koncem řádku. Klíčová slova a názvy funkcí se obvykle píšou malým písmem.

Operátory používané ve Scilabu jsou:

- Aritmetické operátory (+,-,*,/,^)
- Logické operátory (and &, or |, not ~)
- Relační operátory (**==**, **<**, **>**, **<=**, **>=**, **<>**, **~=**)

7.1 Podmínky a cykly

Pomocí cyklů **for** a **while** můžeme vykonávat nějakou činnost opakovaně. Můžeme cyklit buď jeden příkaz, nebo nějaký blok příkazů. Pokud chceme cyklus z nějakého důvodu ukončit, můžeme v bloku uvést příkaz **break**. Cyklus se tak ukončí a program pokračuje dalšími příkazy. Jinou možností je příkaz **continue**. Tento příkaz přeruší vykonávání dalších příkazů v těle cyklu a program skočí na začátek cyklu. [2]

Podmínky umožňují větvení programu.. Jsou to podmínky **if – then – else** a **select – case**.

7.1.1 For

Cyklus **for** provádí příkazy (instrukce) v těle cyklu tak dlouho, dokud platí podmínka.

Syntaxe: **for var = expr (do), instruction, ... , instruction, end,**

kde *var* je proměnná (variable), *expr* je výraz (expression) a *instruction* jsou jednotlivé instrukce (tělo cyklu). Celý cyklus je ukončen příkazem *end*. Příkaz *do* se může vynechat, slouží pouze pro přehlednost celého cyklu.

Příklad: pro *k* od 1 do 4 se vypočítá $x = x \cdot k$

```
--> x = 1;
--> for k = 1:4 do
--> x = x*k
--> end
```

$x = 1$

$x = 2$

$x = 6$

$x = 24$

7.1.2 While

Podmínka v cyklu **while** se vyhodnocuje před provedením cyklu. Pokud platí, cyklus se provede, pokud neplatí, cyklus se ukončí.

Syntaxe: **while expr do instructions,...[,else instructions], end.**

kde *expr* je výraz (expression), *instructions* jsou instrukce, *end* značí konec cyklu *while*. Příkaz *do* se opět může vynechat, slouží pouze pro přehlednost celého cyklu. Ve formě *while do else end* se instrukce umístěná mezi *else* a *end* provede pouze jednou při ukončení smyčky.

Příklad: program vyhodnocuje *n*, dokud je *n* < *2*, pak vypisuje slovo Scilab

```
--> n = 0;
--> while n < 2 do
--> disp ('Scilab');
--> n = n + 0.5;
--> end
  Scilab
  Scilab
  Scilab
  Scilab
```

7.1.3 If – then – end

Syntaxe: **if expr then statements else statements end**

If vyhodnotí logický výraz a pokud je pravdivý, provede následnou skupinu příkazů (*statements*). Podmínka else je volitelná položka. Vyhodnocování podmínek probíhá do té doby, než se některá vyhodnotí jako pravdivá (*true*). Další podmínky se již nevyhodnocují. Pokud se žádná podmínka nevyhodnotí jako *true*, pak se provede tělo bloku za *else*. *Else* se umisťuje vždy na konec.

Příklad: program vyhodnotí, jestli n je nebo není nula.

```
--> n = 1;
--> if n == 0 then
-->   disp ( 'Je nula');
--> else
-->   disp ('Není nula');
--> end;
```

Není nula

7.1.4 Select – case

Syntaxe: **select expr, case expr1 then instructions1, case expr2 then instructions2,..., else instructions, end**

Expr je výraz (*expression*), *instructions* jsou instrukce, *end* značí konec. Na základě výrazu za slovem *select* přeskočí program na návěstí *case* se stejnou hodnotou jakou má výraz za *select* a pokračuje vykonáváním příkazů za ním. Přepínač *select* může obsahovat *else*, na které skočí tehdy, když výraz za *select* neodpovídá žádným návěstím *case*.

Příklad: program vyhodnotí číslo n a vypíše výsledek.

```
--> n = -1;
--> select n
--> case -1.0  disp ('Záporná jednička');
--> case  0.0  disp ('Nula');
--> case  1.0  disp ('Kladná jednička');
--> else       disp ('Jiné číslo');
-->end;
```

Záporná jednička

7.2 Skripty a funkce

Skript je textový soubor, který obsahuje posloupnosti příkazů jako by byly napsány v příkazovém řádku. Obvykle mají příponu **.***sce* a spustit je můžeme příkazem *exec* nebo v nabídce *File -> Exec*.

Do skriptů se můžou zapisovat i komentáře, což slouží pro větší přehlednost. Skript pracuje s existujícími daty nebo může vytvářet nová data nebo proměnné, které mohou být použity pro další výpočty. Pro vytváření skriptů ve Scilabu je vhodné použít editor Scipad.

Funkci je možné definovat přímo v okně Scilabu nebo si ji uložit do souboru (nejčastěji s příponou *.sci*). Funkce jsou malé programy, které přijímají vstupní argumenty a vrací výstupní argumenty.[8]

Definice funkce:

function <output_variables> = <function_name> <function_arguments>

<statements>

endfunction

Function_name je název funkce. *Function_arguments* je seznam vstupních argumentů (proměnných). Tyto argumenty mohou být zapsány ve tvaru *(x1,...,xm)*. Pokud funkce nemá vstupní argumenty, pak se píše **()** nebo nic. *Output_variables* je seznam výstupních proměnných. Mohou být ve tvaru *[y1,...,yn]*. *Statements* je tělo funkce pro příkazy (instrukce).

Vstupními a výstupními argumenty mohou být konstanty, vektory, matice nebo řetězce. Funkce dále může obsahovat komentáře. Pokud funkci definujeme v okně Scilabu a neuložíme ji do souboru, pak funkce po zavření Scilabu nebude k dispozici a musíme ji napsat znovu.

Argumenty funkce (vstupní proměnné) a výstupní proměnné jsou nazývány jako globální proměnné, protože jejich hodnoty jsou dostupné v prostředí Scilabu i po vykonání funkce. Proměnné, které se používají uvnitř funkce a nepatří mezi vstupní a výstupní proměnné, jsou lokální proměnné. Jejich hodnoty nezůstávají přístupné po vykonání funkce.

7.3 Řízení běhu programu

Příkaz **halt** zastaví běh programu. Program pokračuje až po stisku klávesy. Příkaz **pause** zastaví běh programu a přepne do módu *pause*. Tento *pause mód* je uvozen promptem -1->, který indikuje první úroveň *pause módu*. Příkazem **resume** se pokračuje v běhu programu. Příkaz **abort** ukončí běh programu. Pro přerušení běhu programu (např. při zacyklení) slouží klávesová zkratka <CTRL>+<C> . Program zastaví a přejde do *pause módu*. Ukončení se opět provede příkazem **abort**. [9]

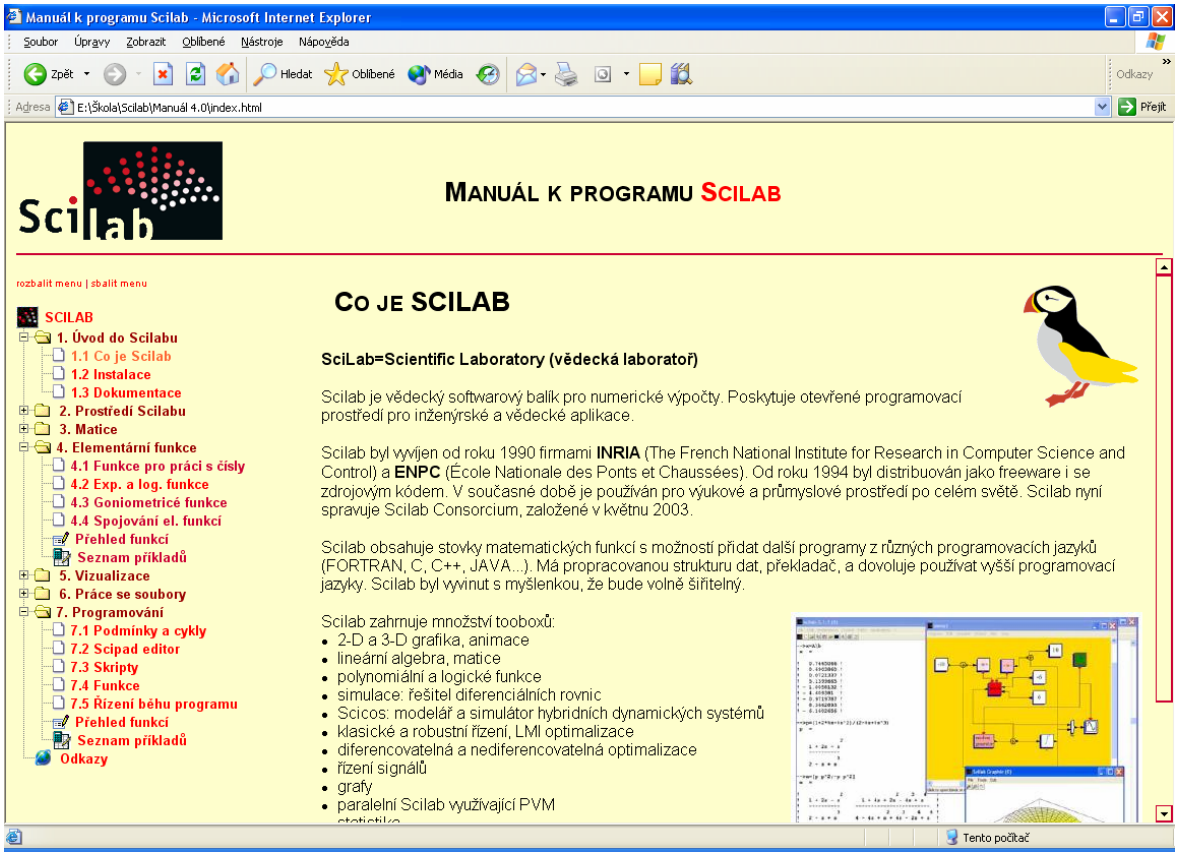
Tyto příkazy jsou také dostupné v nástrojové liště v nabídce *Control -> Resume, Abort, Interrupt*.

II. PRAKTICKÁ ČÁST

8 ELEKTRONICKÝ MANUÁL

8.1 Popis manuálu

Manuál k programu Scilab je vytvořen v prostředí HTML. Pro jednoduchost stránek byly zvoleny rámce. Stránky jsou rámci rozděleny do třech oken. V prvním rámci je záhlaví, kde je umístěno logo Scilabu a titulek. Ve druhém levém rámci je rozbalovací menu stránek, které má charakter stromové struktury. Toto menu je vytvořeno Javaskriptem dTree 2.05 (www.destroydrop.com/javascript/tree/). Je to rozbalovací menu, kdy se po kliknutí na plus rozbalí další položky, podmenu nebo stránky. Kliknutím na mínus se menu zpět zabalí. Třetí okno je hlavní a zobrazuje vybrané stránky. Použité styly, formátování textu a barevné rozvržení je definováno pomocí kaskádových stylů (CSS stylů).



Obrázek 26: Vzhled manuálu

Nad menu jsou přidány dvě pomocné funkce *rozbalit menu* a *sbalit menu*, kdy po kliknutí na tyto položky se celé menu kompletně rozbalí, včetně všech podmenu. Tím vynikne celá struktura manuálu a hlavně tyto funkce usnadní práci. Druhá funkce *sbalit menu* zase celou strukturu zabalí.

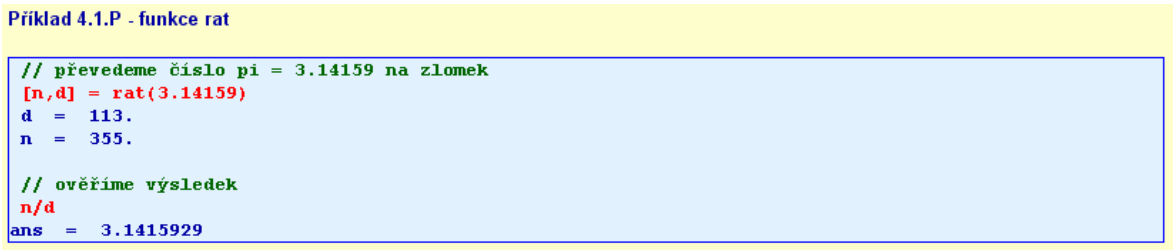
8.2 Struktura stránek

Manuál je rozdělen do sedmi kapitol podle tématu. V první kapitole je úvod do dané problematiky, stručný popis programu a jeho instalace v operačním systému Windows. V druhé kapitole *Prostředí Scilabu* je popsáno základní ovládání programu, popis nápovědy a demonstrací, základní informace pro práci s programem. Třetí kapitola *Matice* uvádí do problematiky práce s maticemi, vektory a polynomy. Ve čtvrté kapitole *Elementární funkce* je vysvětleno použití matematických funkcí, goniometrických funkcí a exponenciálních a logaritmických funkcí. Pátá kapitola *Vizualizace* se zabývá tvorbou 2D a 3D grafů, popisem práce s grafikou a grafickými proměnnými. V kapitole šest *Práce se soubory* je popsán vstup a výstup dat do souboru, binárních i textových. Podrobně jsou popsány funkce jazyka C pro práci se soubory. Poslední sedmá kapitola *Programování* se zabývá vytvářením vlastních funkcí a skriptů, je zde také popis textového editoru Scipad a popis práce s programovacími nástroji jako jsou podmínky a cykly.

Na konci každé kapitoly je přehled všech použitých funkcí se stručným popisem, co daná funkce dělá. Funkce jsou rozděleny kategoricky podle podkapitol nebo jsou seřazeny abecedně. Je zde také seznam uvedených příkladů. Tyto seznamy jsou vytvořeny pomocí hypertextových odkazů, tedy po kliknutí na odkaz se uživatel dostane přímo k požadovanému příkazu nebo příkladu.

Na většině stránek pod hlavním nadpisem je umístěn seznam s odkazy na stránce. Tyto odkazy jsou hypertextové a představují obsah stránky.

Dále je zde množství příkladů, které jsou značeny modrým popiskem a jsou zvýrazněny modrým rámečkem. Text v příkladech je barevně odlišen podle druhu – zelené jsou komentáře, červené jsou příkazy a modré jsou výpisy ze Scilabu.



Obrázek 27: Příklady v manuálu

ZÁVĚR

Internet je obrovským zdrojem volně dostupných programů. Smyslem této práce je ukázat, že i software, který je zdarma, může být velmi dobrou náhradou komerčních programů. Scilab je distribuován jako open source, tedy ke stažení je i se svým zdrojovým kódem. Svou strukturou a funkcemi se nejvíce podobá programu Matlab. Oba programy mají většinu funkcí a příkazů stejných nebo podobných, proto pro uživatele, který ovládá jeden z těchto matematických nástrojů, by neměla být obtížná práce i s druhým programem. Scilab může Matlabu konkurovat v matematických výpočtech, v grafickém prostředí i v rychlosti zpracování příkazů a výpočtů, ale za slabinu lze považovat málo propracovanou a nepřehlednou nápovědu.

Scilab byl původně určen pro operační systém Linux, dnes je dostupný i pro další operační systémy. Může docela dobře sloužit jako vědecká kalkulačka, programovací nástroj nebo vizualizační program pro tvorbu 2D a 3D grafiky.

Úkolem bylo vypracovat podrobný manuál pro dané skupiny funkcí a příkazů v prostředí HTML stránek. Manuál jsem rozdělila do sedmi kapitol podle tématu. Postupně je popsána práce od úplných základů až po složitější operace a náročnější příklady. U každé funkce je uvedena a podrobně vysvětlena syntaxe. Použití funkcí a matematických operací je ukázáno na mnoha příkladech, grafech nebo vytvořených souborech.

SEZNAM POUŽITÉ LITERATURY

[1] INRIA, SCILAB [online]. [cit 2006-5-11]. Dostupný z WWW: <http://www.scilab.org>.

[2] INRIA, SCILAB, *Documentation* [online].[cit 2006-5-12]. Dostupný z WWW: <http://www.scilab.org/download/index_download.php?page=documentation.html>.

[3] INRIA, SCILAB, *Scilab Licence* [online]. [cit 2006-5-12]. Dostupný z WWW: <http://www.scilab.org/legal/license.html>.

[4] INRIA, SCILAB, *Books, Reports, Articles* [online]. [cit 2006-5-12]. Dostupný z WWW: <http://www.scilab.org/publications/index_publications.php?page=books.html>.

[5] INRIA, SCILAB, *Scilab Demonstration Pages* [online]. [cit 2006-5-12]. Dostupný z WWW: < http://www.scilab.org/doc/demos_html/index.html >.

[6] INRIA, SCILAB, *Matlab Scilab Functions* [online]. [cit 2006-5-12]. Dostupný z WWW: <http://www.scilab.org/product/dic-mat-sci/M2SCI_doc.htm>.

[7] INRIA, SCILAB, *FAQ* [online]. [cit 2006-5-12]. Dostupný z WWW: <http://www.scilab.org/legal/index_legal.php?page=faq.html>.

[8] URROZ, Gilberto. *G. Urroz Scilab Webpage* [online]. [cit 2006-4-10]. Dostupný z WWW: <http://www.engineering.usu.edu/cee/faculty/gurro/Scilab.html>.

[9] ROUAULT, Jacques-Deric. *Scilab step-to-step*, CNRS/INRIA [online]. [cit 2006-5-11]. Dostupný z WWW: <http://h0.web.u-psud.fr/orscilab/index.html>.

[10] *Geometrie, matematika* [online]. [cit 2006-4-23]. Dostupný z WWW: <http://www.celysvet.cz/geometrie.php>.

[11] *Matematika online – matice a determinanty* [online]. [cit 2006-5-11]. Dostupný z WWW: <http://math.fme.vutbr.cz/default.aspx?section=63&server=1&article=33>.

[12] DUŠEK, František. *Matlab a Simulink – úvod do používání.* Univerzita Pardubice, 2000. ISBN 80-7194-273-1.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

INRIA	French National Institute for Research in Computer Science and Control.
ENPC	École Nationale des Ponts et Chaussées
GED	Grafický editor
ans	Answer
např.	Například
atd.	A tak dále
id	Identifikační číslo
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: SEZNAM POUŽITÝCH FUNKCÍ

abort
abs
acos
acosh
asin
asinh
atan
atanh
break
browsevar
ceil
clean
continue
contour
contour2d
cos
cosh
cotg
coth
det
diag
diary
disp
exp
eye
file
fix
floor
for
fprintf
fscanf
function
gamma
gca
gce
gcf
gda
gdf
getcolor
getlinestyle
getmark
halt
help
hist3d
histplot
if
input
int
inv
legend
linspace
load
log
log10
log2
logspace
mclose
meof
mesh
mfprintf
mfscanf
modulo
mopen
mprintf
mscanf
msprintf
msscanf
ones
param3d
pause
pie
plot
plot2d
plot2d2
plot2d3
plot2d4
plot3d
plot3d1
plot3d2
plot3d3
poly
printf
rand
rank
rat
resume
roots
round
save
scanf
scf
scipad
select
sign
sin
sinh
size
sprintf
sqrt
sscanf
sum
surf
tan
tanh
trace
while
who
whos
xstring
zeros

PŘÍLOHA P II: CD-ROM

Součástí bakalářské práce bylo vypracování manuálu v prostředí HTML. Soubory jsou dostupné na přiloženém CD-ROM.